Tektronix®
COMMITTED TO EXCELLENCE

TEKniques Vol. 5 No. 4 T1

062-5981-01

DOCUMENTATION

Applications Library
Group 451
Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97007

062-5981-00

10/81

The program material contained herein is supplied without warranty or representation of any kind. Tektronix, Inc., assumes no responsibility and shall have no liability, consequential or otherwise, of any kind arising from the use of this program material or any part thereof.

Prepared by the 4050 Series Applications Library. The 4050 Series Applications Library is maintained as a service for our customers by the Information Display Division of Tektronix, Inc., Group 451, P.O. Box 500, Beaverton, Oregon 97077 U.S.A.

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	PART NUMBER
TEKniques Vol. 5 No. 4 T1	062-5981-01
ORIGINAL DATE: December, 1981 / REVISION DATE:	
AUTHOR	PERIPHERALS

ABSTRACT

TEKniques Vol. 5 No. 4 T1 tape consists of 67 programs on 93 files.

A set of 10 programming aids are included on files 1-32.
- The TAPEMENU routine and its data file occupy the first two files.
- Nine additional programming aid routines, extensively REMarked, reside on files 3-11.
- File 12 contains notes on these programs.
- These same 10 programming aid routines stripped of REMark statements (using program 7) occupy files 13-22.
- Files 23-32 contain the cross-referenced programs using program 8, parse option 1.

Files 33-44 comprise the 2-D and 3-D graphing contest entries, the contest sponsored by TEKniques in the spring of 1981.

File 45 contains the Reference Index program. It must be transferred to a dedicated tape before running. The documentation contains complete instructions for accomplishing the transfer.

Files 46-93 are 47 routines with index printed in the GPIB Programming Guide (part #070-3985-00). These files must be transferred to another tape where they occupy the first 48 files. Instructions to transfer are included.

These routines provide examples of and utilities for a variety of system configurations for the 4050 and TEKTRONIX TM5000 series instruments. The GPIB Programming Guide is not included in this documentation but may be ordered separately by its part #070-3985-00.

Read the documentation for a program before running it.

AUTO LOAD the TEKniques Vol. 5 No. 4 T1 tape, input "1" for device #, then choose a program from the tape menu. NOTE: Leave the TEKniques tape on safe so you won't inadvertently overwrite a file.

TITLE	PART NUMBER
TEKniques Vol. 5 No. 4 T1	062-5981-01

TITLE	PART NUMBER
TEKniques Vol. 5 No. 4 T1	062-5981-01

TITLE	PART NUMBER
TEKniques Vol. 5 No. 4 T1	062-5981-01

TRANSFERRING FILES TO A NEW TAPE

PLOT 50 General Utilities Vol. 1 (TEKTRONIX Part #4050A08) contains a program to transfer any type of 4050 files (program/data/text) quickly and easily along with the header names; however, it requires a 4924 Tape Drive, as does TAPEDUPE program contained on TEKniques Vol. 5 No. 4 T1 tape,

Transferring ASCII or BINARY PROGRAMS without a transfer program

Step 1. Do a TLIST of the MASTER program tape.

Step 2. Record which files go with which program (they are all named) and the size of each file.

Step 3. MARK your new tape to accept the respective files for that program, e.g.,

```
FIND 0
MARK 1,20000
FIND 2
MARK 1,4000
etc.
```

Step 4. Insert the MASTER tape.

FIND a file

OLD for ASCII or CALL "BOLD" for BINARY

Step 5. Insert the new tape

FIND the file to receive the file in memory

SAVE for ASCII or CALL "BSAVE" for BINARY

REPEAT Steps 4 and 5 until all files comprising that program are transferred to the new tape. Note: This procedure will not retain the file header names.

Transferring ASCII or BINARY DATA to a new tape

The 4051R06 Editor ROM could be used to transfer ASCII DATA files.

4050 Applications Library program "Binary Data File Duplicator" will transfer BINARY DATA files without any peripheral.

4050 Applications Library program "Tape Duplication" will transfer ASCII or BINARY DATA or PROGRAM files, but requires a 4924 Tape Drive.

Both of these programs are contained on the 4050 Applications Library UTILITIES T1 tape (TEKTRONIX Part # 062-5974-01), and UTILITIES D1 disk (TEKTRONIX Part #062-5975-01).

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	
TAPEMENU	
ORIGINAL DATE: January, 1979 / REVISION DATE: October, 1980	EQUIPMENT AND OPTIONS REQUIRED: 16K (see below)
AUTHOR: Bob Manthey, County of San Diego, San Diego, CA	PERIPHERALS: Optional - 4924 Tape Drive

ABSTRACT

Files: 1 ASCII Program
1 Binary Data

Statements: 188

This program will establish, maintain, and display a binary encoded data file containing the names and file numbers of all the ASCII Program files found on the tape. (The names are those previously created with 'FMARKER'.) Upon selecting a program by its relative display-number, TAPEMENU will load it and run it.

Memory:

The documented program is -7K in size.

The stripped program is -4K in size.

The binary data file is 1+K in size and will hold 100 program names and numbers.

4924 Option:

Immediately after loading, the program prompts for the drive number (2 = 4924). Pressing RETURN will default to drive #1, the 4050 internal tape unit.

TITLE

TAPEMENU

1. The program is written in an adequately REMarked modular style, and in general, performs as follows:

   1. Select the file drive #.
   2. Validate existence of data file. It assumes the data file to be a Binary Data file residing on file 2.
   3. If file 2 is not a Binary Data file, create a blank data file. (CAUTION: it will automatically MARK file 2.) Jump to 5.
   4. If existent, read the data file.
   5. Sign-on with Tape Name and Identifier, date and device-drive #.
   6. Display programs available with an assigned relative-number.
   7. Prompt for, and validate, a program selection.
   8. If valid, prompt for, and validate, a device drive #.
   9. Load and run (chain) the selected program.
   10. If invalid (or 'RETURN'), prompt for new tape-menu compilation.
   11. If NO, loop to 7.
   12. If YES, compile a new tape-menu data-file.
   13. Loop back to 5.

   NOTE: Modify line 460 to change the blank lines in the display and, hence, the total number of displayable files.

   TO END THIS ROUTINE, PRESS BREAK TWICE.

2. The program creates and maintains one 1+K binary encoded data file which contains data read from the tape-file headers (or extenders):

   T - Total number of programs stored (100 maximum).
   D$ - Date of latest file activity.
   F$ - Composite File #-&-Program-Name string.
   Q$ - Tape-cartridge Identifier (Acronym-#, etc.)
   R$ - Tape-cartridge Name.

   NOTE: Read the WARNING in the program listing-header!!!
   If you wish to change the files storage capacity, change 'T' in line 136.

TITLE

TAPEMENU

3. See program lines 100-199 for Internal Data Storage.

4. See the program 'REMARKs' for the methods used.

5. The program prompts for all required input. Inputting 'RETURN' on the Drive # prompt will default to the 4050 after AUTOLOAD, and to the currently selected drive after Program # input.

   All YES/NO questions may be answered affirmatively with YES, Y or 1, and negatively with any other input (e.g., RETURN).

6. There are no references.

7. The program tape includes a Cross-Reference Index of the program.

FINAL COMMENT:

Any tape cartridge program file names (Identifier, etc.) can be easily modified with the FMARKER utility. The accuracy of the names stored and displayed by TAPEMENU is dependent on that.

Compiling a new or modified data file is relatively easy with this TAPE MENU utility.

The date is taken from the LAST file on a tape. Use FMARKER utility to name the LAST file with a date (if desired).

OPERATING INSTRUCTIONS

Transfer the TAPEMENU program to your new tape, as file 1. Use the TAPEDUPE program to accomplish the transfer since it will save all header information.

Optional-Use FMARKER program to rename the tape cartridge and give it a new number.

RUN the TAPEMENU program; it will initialize a data file on file 2.

As you add programs to your tape, run FMARKER to name them, then RUN TAPEMENU to update the data file on file 2.

When you press AUTOLOAD, TAPEMENU will display the data file and give you the option of choosing a program, or updating the data file.

```
T B A S I C   P R O G R A M   L I B R A R Y   O F

=  TEKTRONIX APPLICATIONS LIBRARY PROGRAMS  =

09-10-81

Tape CARTRIDGE:  #VOL5NO4T1  @ Device #33

==========================================================================
  1- TAPEMENU   2- ReadNote   3- WritNote   4- FMarker    5- LHeader
  6- PHeader    7- Stripper   8- CrossRef   9- TapeDupe  10- CompSort
 11- TAPEME:S  12- ReadNo:S  13- WritNo:S  14- FMarke:S  15- LHeade:S
 16- PHeade:S  17- Stripp:S  18- CrossR:S  19- TapeDu:S  20- CompSo:S
 21- TAPEME:R  22- ReadNo:R  23- WritNo:R  24- FMarke:R  25- LHeade:R
 26- PHeade:R  27- Stripp:R  28- CrossR:R  29- TapeDu:R  30- CompSo:R
 31- Z=F(X,Y)  32- StereoPt  33- W/WoHdnL  34- PlotThrD  35- Append
 36- Append    37- PipeChrt  38- Contours   39- Hierarcl  40- Nonlnear
 41- Vertical
==========================================================================

          Input 'PROGRAM' Option  >
          Compile a 'NEW' TapeMenu  >
```

If you input a number, TAPEMENU will find that program and OLD it into memory. If you press RETURN, you are given the option of creating a new TAPEMENU data file. To do this, TAPEMENU reads all the file headers on the tape, if they are ASCII Programs, it adds that name to its list. It then rewrites the data file with these names.

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
READNOTE		
ORIGINAL DATE	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
January, 1979	October, 1980	8K
AUTHOR		PERIPHERALS
Bob Manthey	County of San Diego San Diego, CA	

ABSTRACT

Files: 1 ASCII Program

Statements: 37

A short utility routine that displays a data file created by WRITNOTE.

However, the core of this routine and its methods, may be used as a subroutine within a complex program, allowing it to display instructions that would be prohibitively large to code in the program.

TITLE

READNOTE

OPERATING INSTRUCTIONS

Load the program through the TAPEMENU or FIND 3, OLD and RUN.

The program as coded, will automatically find file 12 and read the text therein.

To change the file location search, re-code statements 130 and 140.

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	
WRITNOTE	EQUIPMENT AND OPTIONS REQUIRED
ORIGINAL DATE: January, 1979 / REVISION DATE: October, 1980	8K+
AUTHOR: Bob Manthey, County of San Diego, San Diego, CA	PERIPHERALS

ABSTRACT

Files: 1 ASCII Program

Statements: 83

This is a quick and dirty utility which created and edited the notes in file 12. It is a line-oriented utility.

Once the notes are created and stored, they may be modified a line at a time. Modifications include:

- Leaving a line as is
- Changing a line
- Inserting a line
- Deleting a line

The program will automatically MARK a file large enough for the newly created, or modified, text and store it.

TITLE

WRITNOTE

OPERATING INSTRUCTIONS

Load the program through the TAPEMENU or FIND 4 and OLD. Remove the TEKniques tape and insert the tape onto which you will store the text.

Change statement 140 to reflect the offset for the file.

Type RUN, and choose to create a file or modify it.

CREATE

You will be prompted for each line of text. The program won't allow you to enter more than 72 characters into a line. After each line is entered, you will be given a chance to change it.

Start the last line of each page with a Control-C. This informs the program READNOTE to pause after it displays this line.

End the text entry by simply pressing RETURN without entering anything.

At this point, the program will find the file, MARK it, and print all the text you have entered out to it.

MODIFY

The program retrieves text from the file a line at a time and gives you a choice for that line.

Leave as is

Press RETURN or enter Ø. That line will be left as is and the next brought in.

Modify

Discards that line and lets you re-type another to take its place.

Insert

Lets you type a line to be inserted prior to the one displayed. Then re-displays it.

Delete

Deletes the line.

End

Reads in the rest of the text, displays it, until it reaches end-of-file, at which time it will re-MARK the file and save the modified text.

When completed, the program will FIND file 1 (the menu file usually) and OLD it.

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
FMARKER		
ORIGINAL DATE	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
February, 1978	October, 1980	8K
AUTHOR		PERIPHERALS
Bob Manthey	County of San Diego San Diego, CA	Optional-4924 Tape Drive

ABSTRACT

Files: 1 ASCII Program

Statements: 255

Use this routine to mark a tape file header with an 8 character name and also a 30 character Remarks-Extender.

In addition, the tape cartridge may be given a 10 character alpha-numeric identifier and a 48 character name.

A listing facility included in this routine will produce an index of the tape files, and compute and display the file space--used and free, in both a byte and block format, with their respective percentages. (This is for a standard DC300A cartridge only!)

Documented program is 8+K in size.

Stripped program is 6+K in size.

4924 Optional:

This routine can be set to operate on the 4924 by means of a program-prompted input. Inputting RETURN defaults to the 4050 tape drive. The device drive that you're operating on is displayed during the utility's operation.

TITLE

FMARKER

I - Briefly the program functions in the following manner:

1. Set drive-device #.
2. Locate and display the file header.
3. Prompt for, and validate, a header marker.
4. Prompt for, and validate, header Remarks-Extender.
5. If header is changed, write it to the file, then read and display it.
6. Repeat 2-5 until a negative response to prompt @ 2.
7. List index of tape files, if desired.

NOTE: If the file changed is number 1, the program will also prompt for the name of the tape cartridge itself, and a number for it. The TAPEMENU routine will pick up this name and number and include it in the display (see the example following).

II. The program creates no files.

III. See lines 100-199 for internal data storage.

IV. See program REMarks for methods used.

V. The program operation is straightforward as it prompts for the required input.

All YES/NO questions may be answered affirmatively with YES, Y or 1, and in the negative with any other input (e.g., RETURN).

VI. The program tape includes a Cross-Reference of FMARKER.

VII. References:

"TEKTRONIX 4051 Graphic System Reference Manual," pp 7-32 to 34.

"TEKniques," May 1, 1979, Vol. 3 No. 3, page 12:
"Extending the Tape File Header", Ed Sawicki, Tektronix, Inc. Long Island, NY.

TITLE

FMARKER

NOTES

FMARKER writes an 8 character marker beginning at column 25 of the header. Unfortunately, whenever you write/save data to a file, the 4051 writes over columns 25 and 26 with spaces. You may alleviate this problem in your programs by using the routines located in the program LHEADER.

Or, you may change this program so the marker is only 6 characters long beginning at space 27. Modify FMARKER as follows:

```
Lines:  166   DIM M$(6)
        1100  (Change ">" to "  >")  (i.e., add two spaces in front of >)
        1130  If LEN(Z$)>6 THEN 1000
        1420  IF NOT (POS(Z$,"S",1)=1 THEN 1500
              (also change the 3's to 1's in lines 1430, 1500 & 1510)
        1710  (Change Z$ to 6 spaces)
        6100  LET H$=REP(M$,27,6)
        7600  LET M$=SEG(H$,27,6)
```

In addition to not allowing number in the marker, line 1330 also eliminates the "." since its inclusion produces the anomaly of not allowing the 4924 to write/save data to said file (no such problem ensues with the 4051).

Note also that the TEKTRONIX PLOT 50 Utility "4924 Tape-To-Tape Copy Utility" will not transfer file header extender-remarks. I have furnished a modified version of that utility which solves this problem and one or two others.

The routine PHEADER provides an example of how the tape cartridge identifier may be used as part of a BASIC program identifier line.

Final Comment

The index produced by this utility is a convenient means of maintaining a descriptive inventory of programs within your system, along with their location, for ease of retrieval and use.

TITLE

FMARKER

OPERATING INSTRUCTIONS

Load FMARKER through the TAPEMENU or FIND 5, OLD and RUN.

The program will prompt you for the device drive (i.e., 4050 or 4924--note that it assumes the device # of the 4924 to be 2), and for the file # to be changed.

It reads the current header of the file and displays it, then requests for a new marker and/or new remark-extender.

To leave it the same, simply press RETURN.

To input a new marker, key in up to 8 characters, all alphabetic.

To input a new remark-extender, key in up to 30 character, any type.

If file 1 has been chosen, the program will also prompt you for a tape number of 10 characters (any type) and/or a tape name of up to 48 characters (any type).

In the examples following the tape cartridge number is: VOL5NO4T1, the tape cartridge name is: TEKTRONIX APPLICATIONS LIBRARY PROGRAMS.

When you are finished renaming tape headers, the program will prompt for a tlisting.

USING TAPEMENU with FMARKER

Once your tape files are named, you may use TAPEMENU to update your data file of tape names on your cartridge.

Program prompts for file number, new name and remarks-extender:

```
                  Drive 33:  File NUMBER >1
=========================================    ================================
 1      ASCII   PROGRAM TAPEMENU  40        ProgramFile Menu with Chaining
Input  MARKER:  File  1>
                          Input  REMARKS    >
 1      ASCII   PROGRAM TAPEMENU  40        ProgramFile Menu with Chaining
=========================================    ================================
                    (Press 'RETURN' to Continue!)
```

Since it was file 1, it also prompts for a tape cartridge number and tape cartridge name:

```
                              TEKTRONIX APPLICATIONS LIBRARY PROGRAMS

                              TAPEFILE INDEX     VOL5NO4T1  Page- 1
No.     File Type             Marker    Size                Remarks
=========================================    ================================
                                                Input Tape NUMBER  >
      Input Tape NAME >
```

```
                    TEKTRONIX APPLICATIONS LIBRARY PROGRAMS

                              TAPEFILE INDEX     VOL5NO4T1  Page- 1
No.      File Type            Marker    Size                Remarks
=============================================    ================================
                    Drive 33:  File NUMBER >2
=============================================    ================================
 2       BINARY  DATA         ↑MenuDat  5       ProgramFile Numbers-&-Names
Input  MARKER:  File  2>
                              Input  REMARKS   >
 2       BINARY  DATA         ↑MenuDat  5       ProgramFile Numbers-&-Names
=============================================    ================================
                    Drive 33:  File NUMBER >3
=============================================    ================================
 3       ASCII   PROGRAM ReadNote  6            ASCII Data-Note Reading Routin
Input  MARKER:  File  3>
                              Input  REMARKS   >
 3       ASCII   PROGRAM ReadNote  6            ASCII Data-Note Reading Routin
=============================================    ================================
                    Drive 33:  File NUMBER >

                          Want to CONTINUE >n

                          Want a  TLISTing >y
```

TEKTRONIX APPLICATIONS LIBRARY PROGRAMS

TAPEFILE INDEX VOL5NO4T1 Page- 1

No.	File Type		TAPEFILE Marker	INDEX Size	Remarks
1	ASCII	PROGRAM	TAPEMENU	40	ProgramFile Menu with Chaining
2	BINARY	DATA	↑MenuDat	5	ProgramFile Numbers-&-Names
3	ASCII	PROGRAM	ReadNote	6	ASCII Data-Note Reading Routin
4	ASCII	PROGRAM	WritNote	12	ASCII Data-Note Writing Routin
5	ASCII	PROGRAM	FMarker	33	Tape-File Header Marker-Extndr
6	ASCII	PROGRAM	LHeader	12	File-Header Library SubRoutine
7	ASCII	PROGRAM	PHeader	6	Standardized Program Header
8	ASCII	PROGRAM	Stripper	36	Program 'REMARK's Stripper
9	ASCII	PROGRAM	CrossRef	52	Cross-Reference ASCII Programs
10	ASCII	PROGRAM	TapeDupe	45	4924 Tape-to-Tape Copy Utility
11	ASCII	PROGRAM	CompSort	32	Sample Bubl-Sort & Shel-Sort
12	ASCII	DATA	PROGNOTE	15	General Library-Programs Notes
13	ASCII	PROGRAM	TAPEME:S	15	STRP: ProgramFile Menu with Ch
14	ASCII	PROGRAM	ReadNo:S	4	STRP: ASCII Data-Note Reading
15	ASCII	PROGRAM	WritNo:S	8	STRP: ASCII Data-Note Writing
16	ASCII	PROGRAM	FMarke:S	22	STRP: Tape-File Header Marker-
17	ASCII	PROGRAM	LHeade:S	10	STRP: File-Header Library SubR
18	ASCII	PROGRAM	PHeade:S	6	STRP: Standardized Program Hea
19	ASCII	PROGRAM	Stripp:S	19	STRP: Program 'REMARK's Stripp
20	ASCII	PROGRAM	CrossR:S	28	STRP: Cross-Reference ASCII Pr
21	ASCII	PROGRAM	TapeDu:S	32	STRP: 4924 Tape-to-Tape Copy U
22	ASCII	PROGRAM	CompSo:S	17	STRP: Sample Bubl-Sort & Shel-
23	ASCII	PROGRAM	TAPEME:R	3	XREF: ProgramFile Menu with Ch
24	ASCII	PROGRAM	ReadNo:R	3	XREF: ASCII Data-Note Reading
25	ASCII	PROGRAM	WritNo:R	3	XREF: ASCII Data-Note Writing

File Space USED: 510 blocks (130,309 bytes) 42.6%
File Space FREE: 685 blocks (175,611 bytes) 57.4%

(Press 'RETURN' to Continue!)

TEKTRONIX APPLICATIONS LIBRARY PROGRAMS

TAPEFILE INDEX VOL5N04T1 Page- 2

No.	File Type	Marker	Size	Remarks
26	ASCII PROGRAM	FMarke:R	6	XREF: Tape-File Header Marker-
27	ASCII PROGRAM	LHeade:R	3	XREF: File-Header Library SubR
28	ASCII PROGRAM	PHeade:R	3	XREF: Standardized Program Hea
29	ASCII PROGRAM	Stripp:R	5	XREF: Program 'REMARK's Stripp
30	ASCII PROGRAM	CrossR:R	6	XREF: Cross-Reference ASCII Pr
31	ASCII PROGRAM	TapeDu:R	8	XREF: 4924 Tape-to-Tape Copy U
32	ASCII PROGRAM	CompSo:R	3	XREF: Sample Bubl-Sort & Shel-
33	ASCII PROG	Z=F(X,Y)	36	GeneralFuncPlot-Dony Roberts
34	ASCII PROG	StereoPt	18	Z=F(X,Y) by P. R. Tregenza
35	ASCII PROGRAM	W/WoHdnL	28	Z=F(X,Y) by Tschimpke
36	ASCII PROGRAM	PlotThrD	9	Z=F(X,Y) by Kennedy & Noma
37	ASCII PROG	Append	14	Append file for #36
38	ASCII PROG	Append	4	Append file for #36
39	ASCII PROGRAM	PipeChrt	23	2-D w/Shading
40	ASCII PROG	Contours	15	Full Quadratic Model
41	ASCII PROG	Hierarcl	22	Cluster Merger
42	ASCII PROG	Nonlnear	21	Nonlinear Mapping Plot
43	ASCII PROG	Vertical	16	Plot of M Rows in N Dimensions
44	ASCII TEXT	StatData	3	Data File for 41,42,43 above
45	LAST	09-10-81	3	ADD TEKniques Abstract Progs
	File Space USED:	787 blocks		(201,276 bytes) 65.8%
	File Space FREE:	408 blocks		(104,644 bytes) 34.2%

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
LHEADER		
ORIGINAL DATE	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
March, 1979	September, 1980	8K
AUTHOR		PERIPHERALS
Bob Manthey	County of San Diego San Diego, CA	

ABSTRACT

Files: 1 ASCII Program

Statements: 77

The FMARKER utility writes an eight character marker beginning at column 25 of the file header. Unfortunately whenever you write/save data to a file, the 4051 writes over columns 25 and 26 with spaces.

You may alleviate this problem by incorporating the two routines contained in this file into your write/save programs.

TITLE

LHEADER

OPERATING INSTRUCTIONS

Load the program through the TAPEMENU, or FIND 6, OLD and RUN.

Instructions will be displayed on the screen. Basically, you delete lines 1, 3060, then save the rest of the program in memory (lines 3100 - 4900) on a separate tape.

When you are ready to save or write a program or data to a file, initialize the following variables:

```
D =  device #
F =  file #
C$ = CHR(13)
DIM E$(210)
DIM H$(42)
DIM Z$(254)
```

then append this saved program to your current program. Immediately before saving your file or writing/printing your data, do a GOSUB 4000. Immediately after storing your file, do a GOSUB 3000. Your old file header will have been replaced.

CAUTION: Make sure the above variables do not clash with those already dimensioned by your program. That is, you may have to do an INIT before dimensioning the three strings.

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		EQUIPMENT AND OPTIONS REQUIRED
PHEADER		
ORIGINAL DATE January, 1979	REVISION DATE September, 1980	8K
AUTHOR Bob Manthey	County of San Diego San Diego, CA	PERIPHERALS

ABSTRACT

Files: 1 ASCII Program

Statements: 38

This file contains an example of how the tape-cartridge identifier created with FMARKER might be used as part of the BASIC program identifier line.

TITLE

PHEADER

OPERATING INSTRUCTIONS

Load the program through the TAPEMENU, or FIND 7, OLD and RUN.

```
          P R O G R A M     H E A D E R     S A M P L E
====================================================================
90 REM ILGTC-01-TB-006-008 (01-19-79), (1487 Bytes), (02-06-80)
91 REM ***  PHEADER  ***   Program Header sample suitable for Appending
92 REM FOR: Tektronix 4051 Plot 50 Basic
93 REM  BY: Bob Manthey,   County of San Diego, Dept of Transportation
94 REM           ↑         Computer Services, Int. Land Graphics Section
95 REM      (Your Name)    5555 Overland Ave., C.O.C. Building #6
96 REM                     San Diego, Calif. 92123   TEL: (714) 565-5309
97 REM

  F I L E    I D E N T I F I E R    D E S C R I P T I O N
  ====================================================
  ILGTC-01-TB-011-011 (01-19-79), (1413 Bytes), (12-14-79)

  FIATC  N TB PFN TFN  Date of      Total Size   Date of
  idcaa  u ea riu aiu  Creation     of Program   Last Revision
  lerpr  m ks olm plm               to Date
  enoet  b ti geb eeb
   tn r  e rc r e   e
   iy i  r o  a r   r
   fm d    n  m
   i  g    i
   c  e    x
   a
   t
   i
   o
   n

                (Press 'RETURN' to EXIT!)
```

```
T B A S I C   P R O G R A M   L I B R A R Y   O F

=  TEKTRONIX APPLICATIONS LIBRARY PROGRAMS  =

                10-08-80

   Tape CARTRIDGE: #TALTC-01   @ Device #33

=======================================================================
 1- TAPEMENU   2- ReadNote   3- WritNote   4- FMarker    5- LHeader
 6- PHeader    7- Stripper   8- CrossRef   9- TapeDupe  10- CompSort
11- TAPEME:S  12- ReadNo:S  13- WritNo:S  14- FMarke:S  15- LHeade:S
16- PHeade:S  17- Stripp:S  18- CrossR:S  19- TapeDu:S  20- CompSo:S
21- TAPEME:R  22- ReadNo:R  23- WritNo:R  24- FMarke:R  25- LHeade:R
26- PHeade:R  27- Stripp:R  28- CrossR:R  29- TapeDu:R  30- CompSo:R
=======================================================================

          Input 'PROGRAM' Option >
```

TITLE

PHEADER

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
STRIPPER		
ORIGINAL DATE	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
November, 1979	March, 1980	32K
AUTHOR		PERIPHERALS
Bob Manthey	County of San Diego San Diego, CA	Optional -4924 Tape Drive

ABSTRACT

Files: 1 ASCII Program

Statements: 260

This program strips the 'REMarks' from an ASCII Program file. One of three stripping modes is available:

1. Delete REMark commentary
2. Delete REMark line
3. Delete REMark line and RENumber the following line

The stripped program is saved and labeled as a separate ASCII Program file. The original program remains intact.

32K Memory: The documented program is 8.5K in size.
The stripped program is 5K in size.

In a 32K single-tape system, the utility will strip a program whose final size can reach 25K maximum. In a dual-tape system, there is no memory limit as the stripped program is transferred to its file on a line-by-line basis.

Optional 4924 Tape Drive: If a 4924 is available, change line 116 to read D2=2 (2 is the device address). The program will automatically compensate.

TITLE

Ascii Program Remarks Stripper Utility

I. The program is written in a modular format, in a top down style with liberal use of 'REMARK's. (I treat the 'REMARK' line of a module as if it were the Label to that routine. You will notice in the stripped version of the program that the first statement-line of each module has been renumbered to that of the former 'REMARK' line of the documented version. This is accomplished using Mode-3 of the 'STRIPPER' utility.) Briefly, the program performs in the following manner:

1. Locate and validate a user selected source file.
2. Load a program line from the file.
3. If a 'LET' statement, delete the 'LET'.
4. If a 'REMARK' line, strip according to the user selected mode.
5. Save line in secondary file if dual-tape system, or add line to Program-Dump$ if single-tape system.
6. Repeat 2-thru-4 until Source File is exhausted.
7. If a single-tape system, create a file and save the stripped Dump$.
8. Mark the new File-Header appropriately.
9. Loop back to 1.

NOTE:

The program automatically sets a file offset for a used "Scratch" tape, only once, after the initial Program Sign-On. This routine is then deleted by the program. (In a single-tape system the "Scratch" tape is the Source tape.) Hence, if you change the "Scratch" tape (or Source tape in single-tape systems) it will be necessary to either relaod the 'STRIPPER' program and start fresh, or change the existing file offset by aborting the program, typing "O2=(new file offset#)", and then rerunning the program. If you wish, you may retain the Offset routines by deleting lines 862 & 864. Of course this will diminish the maximum stripped-program size that a single-tape system may accomodate. The stripped version of this program has stripped a 29k Source program producing a 15k Program-Dump$ using Mode-1 (strip commentary only).

TITLE

Ascii Program Remarks Stripper Utility

Also, Note that Routine 25ØØ modifies an existing file header-extender and truncates it to the 'F Marker' format of 31 Bytes. If you wish to retain the complete extender disable Line 253Ø.

II The Program creates and stores one Ascii File per Source File. If you use a Single-Tape System there will have to be enough free space on the Source Tape to accommodate this file!

III See Lines 1ØØ - 199 for Internal Data Storage.

IV See Program 'Remark's for methods used.

V Program operation prompts for all required input. Inputting 'Return' on Starting Line# will default to the value stored in L1 (see Line 128). 'Return' on Type of Stripping will default to Type-1.

All Yes/No questions may be answered in the affirmative with 'Yes', 'Y', or '1' and negatively with any other input ('Return').

VI There are no references.

VII The Program include a Cross-Reference Index of the "Stripper' Program.

FINAL COMMENT:

This Utility is intended as an aid in programming in that it should allow the programmer to prepare a well documented source program while at the same time producing a more optimally sized run-time program from the source.

The three modes available, 1. Delete REM Commentary,
2. Delete REM Line, or
3. Delete REM Line and Renumber Following Line,

should cover the majority of programming-styles. Mode-1 should be used whenever possible while a program is being debugged as it will allow the programmer to follow the modular structure of the program while eliminating

TITLE

Ascii Program Remarks STRIPPER Utility

the majority of excess verbiage.

WARNING: Mode-2 should never be used on programs in which 'Remark' Lines serve as Labels to Routines within the Program! (This includes all my Programs.)

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
CROSSREF		
ORIGINAL DATE	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
December, 1979	September, 1980	32K
AUTHOR		PERIPHERALS
Bob Manthey	County of San Diego San Diego, CA	Optional - 4924 Tape Drive

ABSTRACT

Files: 1 ASCII Program

Statements: 404

This program will sequentially index a cross-referenced ASCII program. The cross-referencing may include six or 13 commands:

OF, GO TO, GOSUB, THEN, RESTORE, & USING only

or all of the above plus

DELETE, LIST, SAVE, APPEND, RENUMBER, CALL and RUN

The index is saved and labeled as an ASCII program of REMark statements. This file may be loaded later for display, or appended to the source file from which it derived.

32K Memory: The documented program is 15K in size.
The stripped program is 9K in size.

In 32K the program will handle 400 references. Users may experimentally try it in smaller memories by changing the vector R1 and S1 dimensions in line 220-240. NOTE: Lines 1070 - 1090 delete those portions of the program used only once to save space.

Optional 4924 Tape Drive: If a 4924 is available, change line 166 to read D2=2 (where 2 is the device address). The program will automatically compensate.

TITLE "CROSSREF":

ASCII PROGRAM CROSSREFERENCE UTILITY

I. The program is written in a modular format in a top-down style with liberal use of 'Remarks'. (I treat the 'Remark' line of a module as if it were the lable to that routine. You will notice in the stripped program that the first statement-line of each module has been renumbered to that of the former 'Remark' line. This is done automatically by the 'Stripper' program.)

Briefly, the program functions in the following manner:

1. Locate and validate a user selected source file.
2. Load a program line.
3. Parse for 'Reserved-Word's.
4. If found save source-line # & reference-line #.
5. Repeat 2-4 until source file is exhausted.
6. Perform a simultaneous sort of source-line # & reference-line #.
7. Construct a cross-reference index in a 'Remark' line format.
8. Create a file & save the index.
9. Mark the new file-header appropriately.
10. Repeat 1.

NOTE:

The program automatically sets a file offset for a used 'Scratch' tape, once, after the initial program sign-on. This routine is then deleted by the program. (In a single tape system the 'Scratch' tape is the source tape.) Hence, if you change the 'Scratch' tape (or source tape in single tape systems) it will be necessary to either reload the 'Crossref' program and start fresh or change the present file offset by aborting the program, typing '02 = (New File Offset #)' and then rerunning. If you wish you may retain these routines by disabling line 1080 (make it a 'Remark' line). This will naturally diminish the maximum cross-reference index size. The present stripped program has successfully cross-referenced 25K source programs with resultant indexes of 17Ø+ source lines & 25Ø+ reference lines!

TITLE "CROSSREF"

ASCII PROGRAM CROSSREFERENCE UTILITY

Also, note that routine 54ØØ modifies an existing file header-extender and truncates it to 31 bytes. (This fits the 'F Marker' display format.) If you don't wish to truncate the header-extender disable line 552Ø.

II. The Program creates and stores one ASCII file per source file. If you use single-tape system there will have to be enough free space on the source tape to accommodate this file. In effect, the file created is an addendum to the program-source file from which it was derived and may be appended to it if you wish.

III. See Lines 1ØØ - 299 for internal data storage.

IV. See Program "Remark's for methods used.

V. The program operation is straightforward as it prompts for the required input. Inputting 'Return' on starting line # will default to that stored in L1 (see Line 178). 'Return' on type-total of word parse will default to type 1.

All Yes/No questions may be answered affirmitively with 'Yes', 'Y', or '1' and in the negative with any other input ('Return').

VI. There are no references.

VII. Program listings include a cross-reference index of the 'CROSSREF' program.

FINAL COMMENT:

The index produced by this utility is intended as a primary programming tool in that it will quickly allow the programmer to more readily visualize the flow of a program while simultaneously alleviating the potential problems arising from the arbitrary relocation or deletion of program segments.

TITLE

CROSSREF

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
TAPEDUPE		EQUIPMENT AND OPTIONS REQUIRED
ORIGINAL DATE	REVISION DATE	
	October, 1980	16K
AUTHOR		PERIPHERALS
Modified by Bob Manthey	County of San Diego San Diego, CA	4924 Tape Drive

ABSTRACT

Files: 1 ASCII Program

Statements: 320

This program transfers any type of program or data files from the 4050 internal tape drive to the 4924 tape drive. The duplicated file will retain the original header and extender.

The program will automatically MARK the new files, if desired.

TITLE

TAPEDUPE

OPERATING INSTRUCTIONS

Insert the target tape into the 4924 tape drive.

Load the TAPEDUPE program into 4050 memory through the tape directory, or FIND 10, OLD and RUN.

Insert your source tape into the 4050 memory and respond to the prompts.

If you are duplicating the entire tape, the program will automatically assign the source files to the same file numbers on the duplicate tape.

However, if you are duplicating only a portion of the tape, you may choose the file numbers on the 4924 into which to transfer your source files.

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
COMPSORT		
ORIGINAL DATE	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
October, 1980		16K
AUTHOR	County of San Diego	PERIPHERALS
Bob Manthey	San Diego, CA	

ABSTRACT

Files: 1 ASCII Program

Statements: 172

This sample program generates a list of random and ordered numbers which are then sorted by means of a Bubble-Sort routine utilizing a floating window; followed by a Shell-Metzner Sort for comparison. Both routines may be used, as is, with only the array name and element variables changed to ones appropriate to your program.

I. The Bubble-Window sort is my transcription of a North Star Basic routine written by Paul T. Brady, and published in the September 1980 issue of "BYTE" magazine (see attached article).

The Shell-Metzner sort is my transcription of a routine originally transcribed into North Star Basic by Steven Fisher, P.O.Box 457, La Mesa, CA 92041, and made available by him to the San Diego North Star User Group Library in 1978. (See "CROSSREF" for a 2-vector sort modification of the same routine.)

NOTE:

As written, each routine sorts a vector in a low-to-high order. However, the sort order may be changed to high-to-low order by simply changing the vector comparison operator:

```
2330 IF NOT(A(J)< A(J+1)) THEN 2700     ;Bubl-Sort
3410 IF NOT(A(H)< A(L)) THEN 3600       ;Shel-Sort
```

TITLE
"COMPSORT"

Sample: Bubble-Window & Shell-Metzner Sorts

II. There are no data-files involved.

III. See program lines 100-199 for internal data storage.

IV. Briefly, the program performs as follows:

1. Sign-On.
2. Display program description.
3. If requested, call sort-demo routine.
4. Input, and validate, number of random #'s to sort.
5. If "0", jump to 9.
6. Generate and display random and ordered number list.
7. Sort list with Bubble-Window technique and display.
8. Sort list with Shell-Metzner technique and display.
9. If requested, loop back to 4.

See program 'REMARK's for methods used.

V. The program operation is straightforward as it prompts for the required input. Any YES/NO questions may be answered affirmatively with 'YES', 'Y', or '1', and negatively with any other input (e.g. 'RETURN').

VI. See "Sorting With a Catch", "BYTE", September 1980, pp. 322-3, by Paul T. Brady, Marcshire Dr., Middletown, N.J. 07748.

VII. The program listing includes a cross-reference index of the COMPSORT program.

TITLE
"COMPSORT"

Sample: Bubble-Window & Shell Metzner Sorts

FINAL COMMENT:

The Bubble-Window routine "...is designed specifically to sort lists with only a few entries out of order. It can even be used to check a list quickly to ensure that all entries are ordered..."

The Shell-Metzner technique, on the other hand, is reasonably efficient when used with moderate amounts of random sequential data needing to be sorted.

Either routine should be easy to adapt to specific sorting configurations (see "CROSSREF") in a variety of programming languages, or dialects.

30 ELEMENT VECTOR

SORT SPEED COMPARISONS (seconds)

		Bubl-Sort	Shel-Sort	Bubl/Shel %
Random #'s to Sort -	0	3	14	21
	1	7	17.5	40
	15	34	20	170
	30	45	23	193.5

Programming Quickies

6502 Loop Control

Gordon Campbell, 36 Doubletree Rd, Willowdale, Ontario, Canada

For clarity, the best way to loop through a field is to start at the beginning and stop at the end. It is important to be able to change the content or length of the field without having to change the code that handles it. Some people use a *marker byte* such as hexadecimal 00 to stop the loop; however, if you make your assembler work for you, this is unnecessary.

Listing 1 is an example of how to make your assembler perform this task. The X register is used to index through a field. The code is set up so that when the register hits zero, execution is terminated. Thus, begin by loading the register with 256 minus the length of the field. Then work through the field from start to end by loading the accumulator with the byte stored at the end of the message minus 256, plus the contents of the X register. The result is that when the X register hits zero, you are done.

The code shown has been used with two assemblers: Carl Moser's ASSM/TED, and Dan Fylstra's *6502 Assembler in BASIC*, published by Personal Software. Fylstra's assembler generates an error message on the first pass if MSG and MSGEND follow the code that uses them, but then produces correct object code. Of greater concern is the fact that both assemblers do not notice if MSG is greater than 256 bytes long. This should be an error condition that raises a diagnostic. In both cases the only result is that incorrect code is produced.

```
                        0010            .BA $7000
                        0020            .OS
                        0030            .LS
                        0040 ; ** HOW TO SCAN A FIELD **
                        0050 ; (MAKE YOUR ASSEMBLER WORK)
                        0060 ;
                        0070 ; THE OPTIMUM METHOD OF LOOP
                        0080 ; CONTROL ON A 6502. MAXIMUM
                        0090 ; OF 256 BYTES OF DATA.
                        0100 ;
                        0110 ;          .
                        0120 ;          .
                        0130 ;          .
                        0140 ;          .
7000- A2 F1             0150            LDX #MSG+256-MSGEND
7002- BD 1B 6F          0160 PRLOOP     LDA MSGEND-256,X
7005- 20 D2 FF          0170            JSR PRINT
7008- E8                0180            INX
7009- D0 F7             0190            BNE PRLOOP
                        0200 ;          .
                        0210 ;          .
                        0220 ;          .
700B- 00                0230            BRK
                        0240 ;
700C- 50 4C 45          0250 MSG        .BY 'PLEASE PRINT ME'
700F- 41 53 45
7012- 20 50 52
7015- 49 4E 54
7018- 20 4D 45
                        0260 MSGEND
                        0270 ;
                        0280 PRINT      .DE $FFD2
                        0290            .EN

LABEL FILE:  [ / = EXTERNAL ]

PRLOOP=7002             MSG=700C        MSGEND=701B
/PRINT=FFD2
//0000,701B,701B
>
```


Sorting With a Catch

Paul T Brady, 91 Marcshire Dr, Middletown NJ 07748

So much has been said concerning various sorting algorithms that it hardly seems possible to be able to contribute to this topic; and yet, in a small business (a nature

center, to be precise), we have developed a sorting routine that handles accounting entries, mailing list entries, etc, at a speed that leaves fancy algorithms in the dust. The special beauty of this technique is that it is very simple, and involves only a slight modification of the usually terribly inefficient brute-force *bubble* technique.

The routine has another advantage—it will not disturb the order of ties. For example, if one orders by zip code, it will not rearrange entries having the same zip code. This is an advantage if the list were previously alphabetized and you wanted to retain alphabetization within zip codes.

There is a catch. This routine is absolutely terrible for ordering a true random list. The routine is designed to handle a list that already is nearly in order, and you want to add a few extra items. But this is exactly the case in a mailing list, in which you add 20 names to a 1500-name list, or in accounting, in which you add 15 transactions to a 60-item list.

The Algorithm

The algorithm works as follows: assume that you have an array of L items, A(I), I = 1 to L. In the standard bubble sort, you compare A(1) with A(2). Assume that you want the list ordered from smallest to largest entry. Then, if A(1) < = A(2), leave them alone, but if A(1) > A(2), reverse them and proceed pairwise down the list. The last comparison made is between A(L−1) and A(L), reversing them if A(L−1) > A(L). You have just made L−1 pairwise comparisons.

For those unfamiliar with this method, a moment's thought should demonstrate that in this first pass you have guaranteed that the largest entry has sunk to the bottom. That is, A(L) now is the largest entry. In subsequent passes, it is no longer necessary to test anything against A(L). So, the second pass ends by comparing A(L−2) with A(L−1). But now, you have guaranteed that the second biggest entry is in the L−1 slot, so each successive pass requires one less comparison.

Even with the shortcut of cutting each pass to be one shorter than the previous pass, this method still takes a long time. But now consider the following. Suppose, during the first pass of L−1 comparisons, we check to see just how well ordered the list already is. We will set up a *window* in which W equals the first pair that was ordered, and X equals the last pair. Suppose the list contains 85 items, but after the first pass, W = 26 and X = 34. This means that everything beyond 34 is already ordered. Items earlier than 26 may not be completely in order when considering later items, but the very next pass can compare entry twenty-five with entry twenty-six; ie: at W−1. So, we have a window that will ascend to the top of the list. Further, on each successive pass we will reevaluate W and X. As soon as X < = 1, we can stop. (Note: X can equal zero in the special case that the entire list was already in order before you invoked the routine.)

The Program

This idea is so simple that it cannot be new; yet, I have not seen it mentioned, and even if it is published elsewhere, it is worth repeating. The code in listing 1 is for North Star BASIC, in which the semicolon separates statements on the same line. W and X have already been defined. T, T1, and T2 are temporary variables. I is an index variable, and A(I) is the array. The A(I) could also be pointers to string variables; the technique is clearly not limited to ordering numbers.

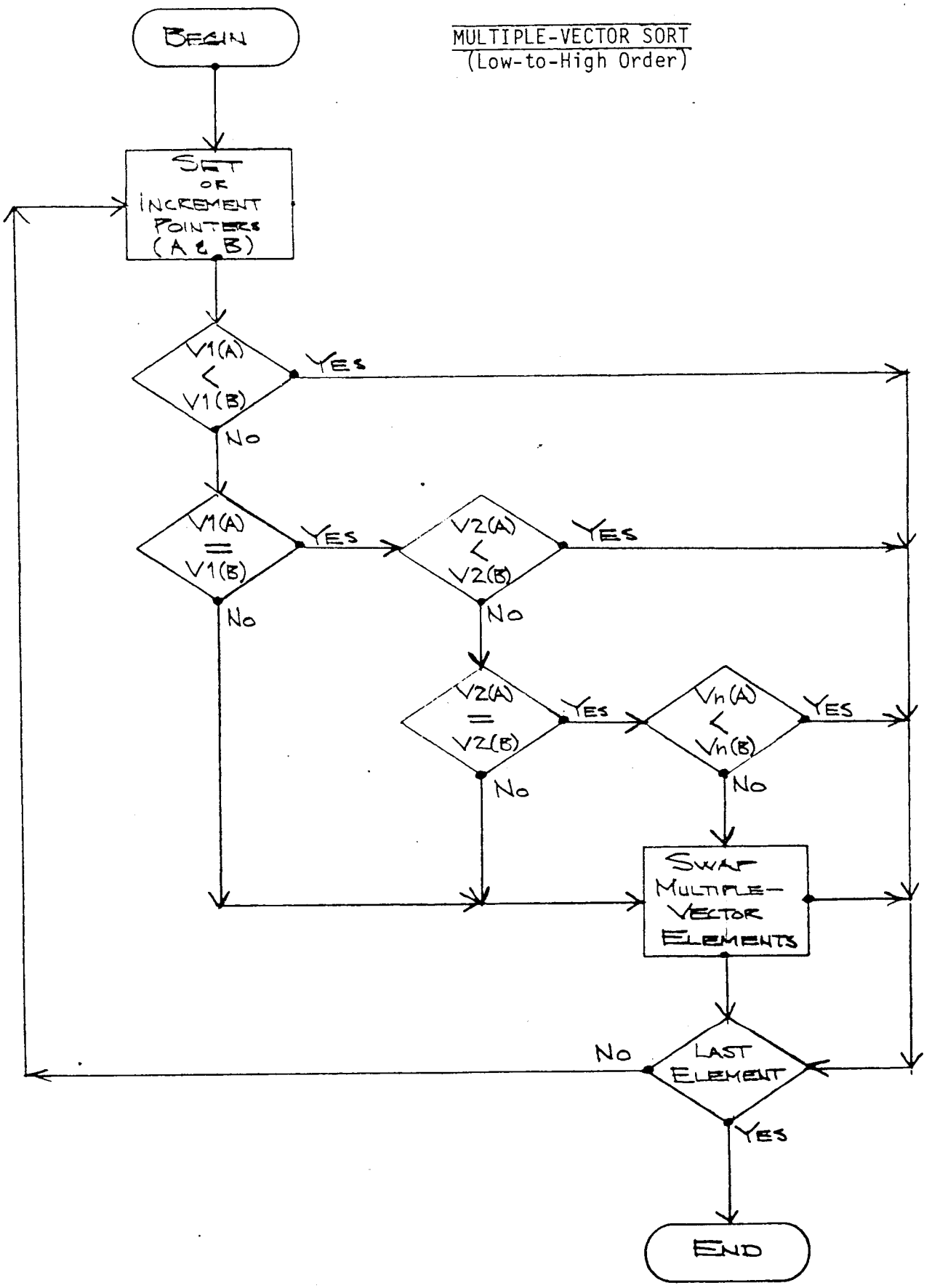
A final comment. This routine is at its very best if the list is already completely ordered before calling it; it makes one pass through the list, discovers that the list is already ordered (X=0 in statement 135), and quits. This is not at all a ridiculous situation. We have several programs that require ordered data in files, and call the sort routine whenever a "write" is called for, even if nothing was done to disturb the order. In such instances, the sort is only a momentary delay.

Listing 1: *A bubble sort with a window. This routine is designed specifically to sort lists with only a few entries out of order. It can even be used to check a list quickly to ensure that all entries are ordered. The main attraction, though, is its simplicity; the actual North Star BASIC code is only eight lines long.*

```
100 W=2;X=L;REM W=UPPER WINDOW BOUND, X=LOWER
105 FOR I=1 TO L
110 T1=X;X=0;IF W<2 THEN W=2;T2=W-1;W=0
115 FOR J=T2 TO T1-1;REM BEGIN AT T2. STMT 110
    ASSURES T2>=1.
120 IF A(I)<=A(I+1) THEN 135
125 T=A(I);A(I)=A(I+1):A(I+1)=T;REM. OUT OF ORDER,
    REVERSE.
130 X=J;IF W=0 THEN W=J; REM W=0 IMPLIES FIRST
    REVERSAL.
135 NEXT;IF X<=1 THEN EXIT 145;NEXT
140 STOP;REM FOR COMMENT ONLY - WILL NEVER BE
    REACHED.
141 REM WILL NEVER FINISH SECOND "NEXT" OF 135
145 REM ROUTINE ENDS HERE, LIST IS ORDERED. ■
```

TITLE

COMPSORT



Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	
General Function Z=F(X,Y) Plot	
ORIGINAL DATE: February, 1981 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 32K
AUTHOR: Dony Robert Brussels, Belgium	PERIPHERALS: Optional - 4662 Plotter

ABSTRACT

Files: 1 ASCII Program

Statements: 452

This program draws two variable functions, z=f(x,y), with hidden lines removed. The draw is made in a rectangular region so that X1≤X≤X2 and Y1≤Y≤Y2.

The function may be drawn on a block representing axes parallel to the three real axes X,Y,Z as well as the study intervals.

The user writes the function to be studied, for instance in the form:

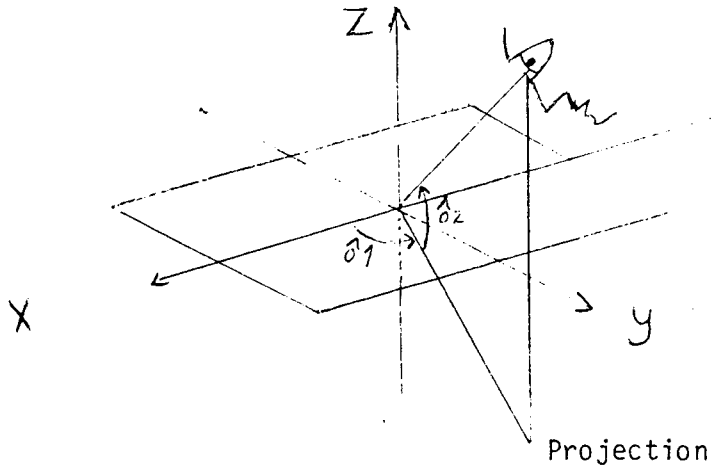
2500 Z=-8*EXP(-X*X-Y*Y)*(X+Y)

As soon as the user types RUN (RETURN), the program asks the inputs necessary to do the calculations and prompts for the following options.

1. Any seeing angle. The eye always looks at the origin of the axis. The azimuth angle O1 can vary from 0^{o} to 190^{o}, and from 0^{o} to -180^{o}. The dip angle O2 can vary from 0^{o} to 90^{o} and from 0^{o} to -90^{o}.
2. The number of slices cutting the surface.
3. The number of joints on each slice.
4. The eventual file number to record the drawing in G.D.U.s that allows a fast redraw of the function later.
5. The true scale or a uniform scale.
6. One or two directions of cutting.
7. The drawing of the function with or without the axis block.

TITLE

General Function Z=F(X,Y) Plot



Projection of the eye on the XY plane

OPERATING INSTRUCTIONS

Load the program into memory through the tape directory, or FIND33, and OLD. Change statement 2500 to the function of your choice, then RUN the program and respond to the prompts.

TITLE

General Function Z=F(X,Y) Plot

COMMENTS

I have tried to scan always the screen always from right to left (climbing or descending). Points in the foreground are always drawn first. After the drawing of each line, the maximal and minimal crest lines are update. Any line situated inside these two crest lines is eliminated. Moreover the program also draws the edges of the surface when two directions are asked.
There are only 8 different cases according to the choice of the angles O1 and O2. They are called ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ . They are reduced to two cases only for each direction. The increments for the lines and the points are Y3 and X3.

G1 = 1 or 2 for the first direction
G1 = 3 or 4 for the second direction.

Let us examine two cases among the eight! See the following page 4 .

Case ① : first direction G1=1 (index loop G=1)

- the fist line (FL1) is Y=Y2 and starts from X=X1
- the following lines are obtained by calculating Y=Y2-(U-1).Y3
- the following points on each line are calculated by X=X1+V.X3

second direction G1=3 (index loop G=2)

- the first line (FL2) is X=X2 and starts from y=Y2
- the following lines are obtained by calculating X=X2-(U-1).X3
- the following points on each line are calculated by Y=Y2-V.Y3

Case ⑧ : first direction G1=2 (index loop G=1)

- the first line (FL1) is Y=Y1 and starts from X=X2
- the following lines are obtained by calculating Y=Y1+(U-1).Y3
- the following points on each line are calculated by X=X2-V.X3

second direction G1=4 (index loop G=2)

- the first line (FL2) is X=X1 and starts from Y=Y1
- the following lines are obtained by calculated X=X1+(U-1).X3
- the following points on each line are calculated by Y=Y1+V.Y3

Reasoning in the same way for the other cases, we easily obtain the following flowchart.(page 5)

TITLE

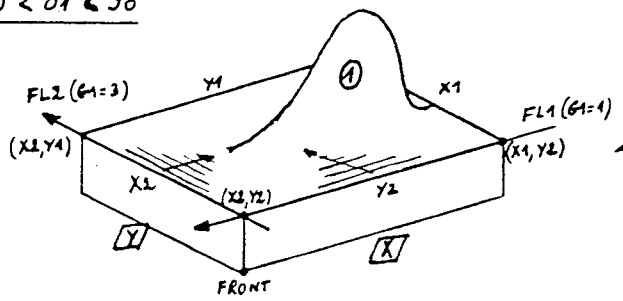
General Function Z=F(X,Y) Plot

$\hat{O2} > 0$: SEEN FROM ABOVE

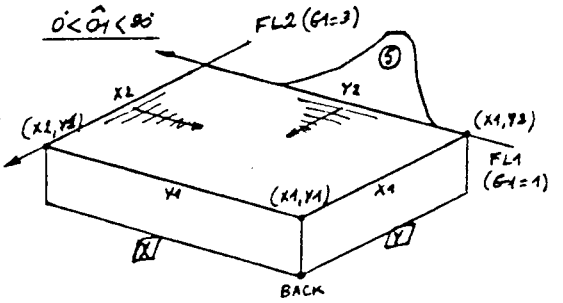
$\hat{O2} < 0$: SEEN FROM BENEATH

2.

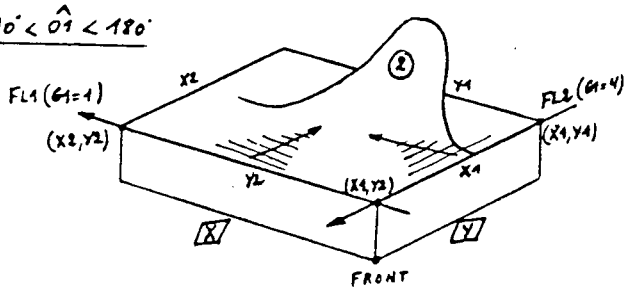
$0° < \hat{O1} < 90°$



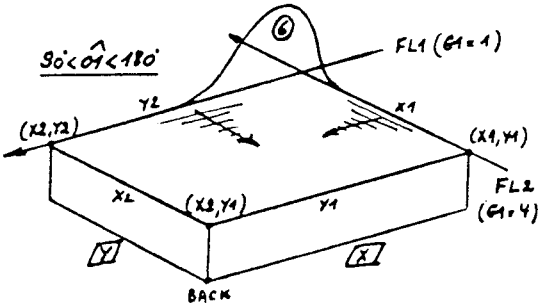
$0° < \hat{O1} < 90°$



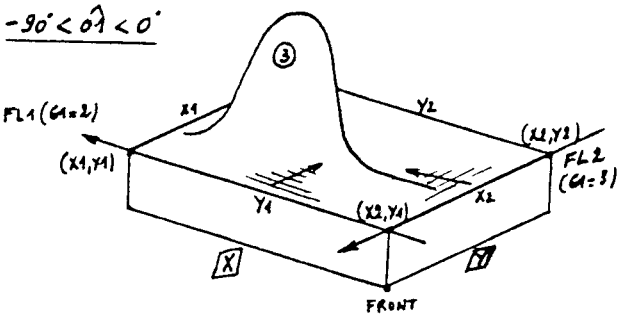
$90° < \hat{O1} < 180°$



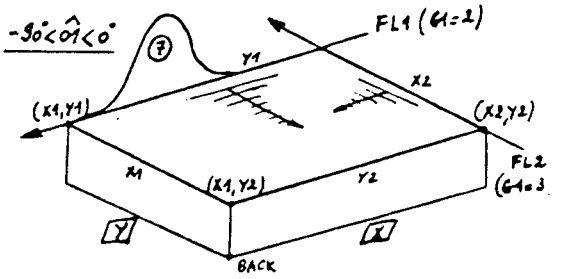
$90° < \hat{O1} < 180°$



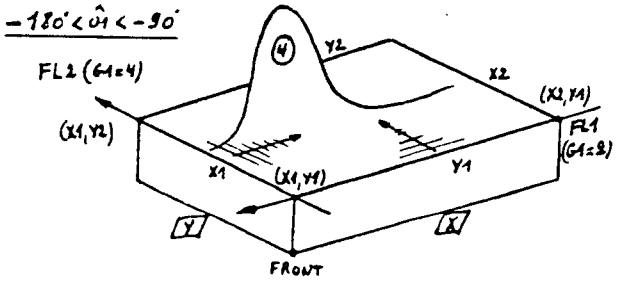
$-90° < \hat{O1} < 0°$



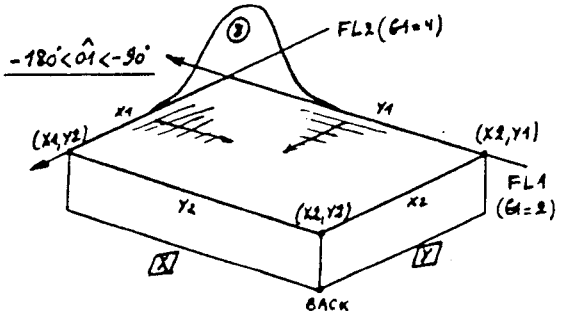
$-90° < \hat{O1} < 0°$



$-180° < \hat{O1} < -90°$



$-180° < \hat{O1} < -90°$



FL1 = FIRST LINE 1ST DIRECTION

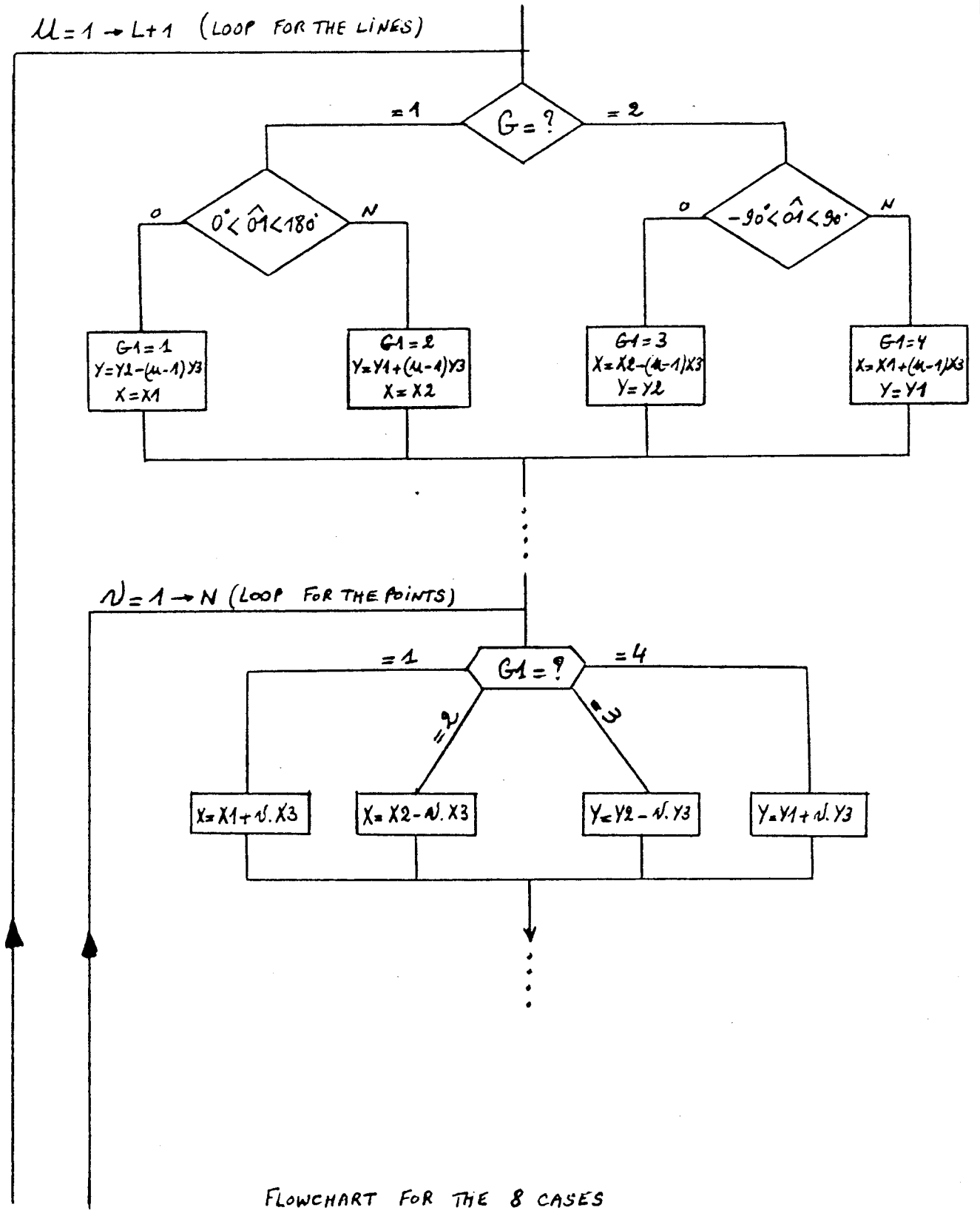
[X] // TO THE X AXES

FL2 = FIRST LINE 2ND DIRECTION

[Y] // TO THE Y AXES

TITLE

General Function Z=F(X,Y) Plot



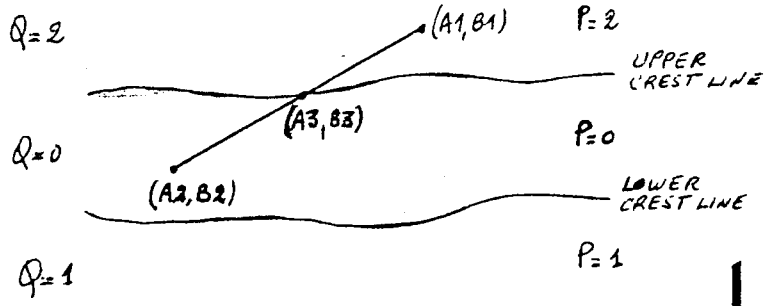
FLOWCHART FOR THE 8 CASES

TITLE

General Function X=F(X,Y) Plot

THE HIDDEN LINE PROBLEM.

Let's take a point (A1,B1) which allows to determine P and the next point (A2,B2) which determine Q. We can have:

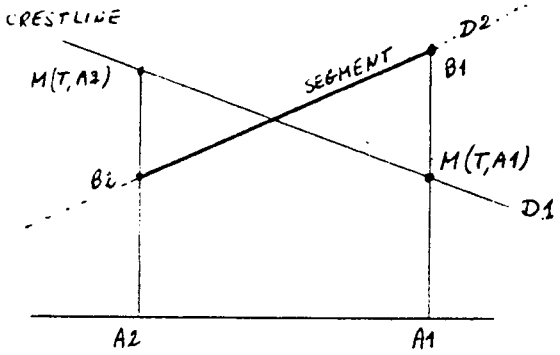


1) P=2 and Q=2 or P=1 and Q=1

That means that the segment joining the two points is visible.

2) P=2 and Q=0 or P=1 and Q=0 or P=0 and Q=1 or P=0 and Q=2

That means that the segment is partly hidden and cuts one of the crest lines. So the intersection point (A3,B3) is to be calculated. It is the variable T which allows to know which crest line is cut. To find (A3,B3), we only have to find the intersection of the two straight lines D1 and D2.



$$D1 \equiv \frac{y - M(T,A1)}{M(T,A2)-M(T,A1)} = \frac{x - A1}{A2 - A1}$$

$$D2 \equiv \frac{y - B1}{B2 - B1} = \frac{x - A1}{A2 - A1}$$

The solution of the system is

$$\begin{cases} A3 = \dfrac{B1.A2 - B2.A1 - M(T,A1).A2 + M(T,A2).A1}{D} \\ B3 = \dfrac{B1.M(T,A2) - B2.M(T,A1)}{D} \end{cases}$$

$$D = M(T,A2) - M(T,A1) - B2 + B1$$

Remark: If D=0, the segment is parallel to the crest line and must be drawn.

3) P=0 and Q=0

Then the segment is completely hidden.

4) $P \neq Q \neq 0$

Then the segment cuts the two crest lines simultaneously. So two intersections points (A3,B3) are to be found. It is the switch variable S which allows, after calculating the first intersection point, to calculate the second one.
If P=1 then T=1 first then T=2.
If P=2 then T=2 first then T=1.

TITLE

General Function Z=F(X,Y) Plot

UPDATING THE CREST LINES MATRIX.

The linear interpolation is used therefore.

We have : $y=m.x + p$

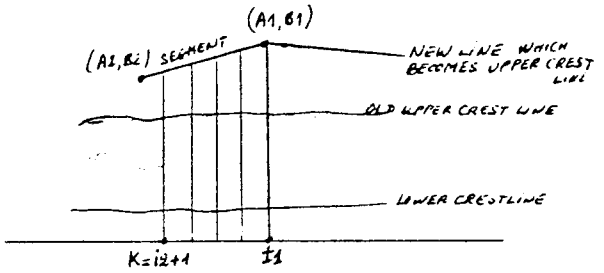
At (A1,B1) → $B1 = m.A1 + p$

At (A2,B2) → $B2 = m.A2 + p$

The solution of that sysytem gives

$$m = \frac{B2 - B1}{A2 - A1}$$

$$p = \frac{B1.A2 - B2.A1}{A2 - A1}$$



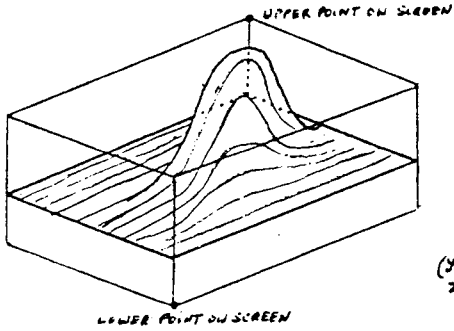
The variables I1 and I2 are the rounded values of A1 and A2.
In a loop with the index K varying from I2+1 to I1 by step 1, we can calculate the corresponding ordinates of the segment with the formula:

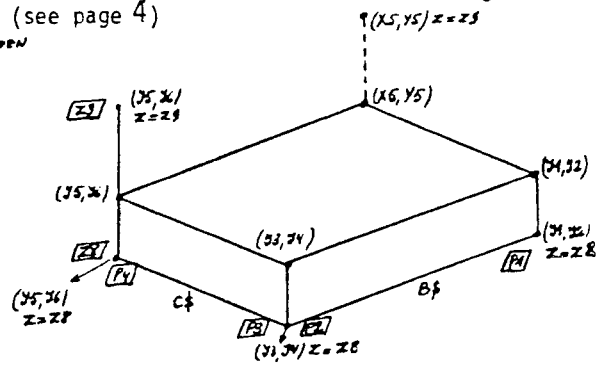
$$R = \frac{K.(B2 - B1) + A2.B1 - A1.B2}{A2 - A1}$$

where K is substitued for x and R for y.

DRAWING OF THE AXIS BLOCK.

We always try for the axis block never to come on the drawn surface except in some cases where it is impossible to avoid. For instance for the function Z=X.X+Y.Y where O1=45 and O2=30 . (see in the lot of drawing)
The representation of the axis block in the 8 cases will always be copied from the following scheme. You only have to see the lowest point on the screen as being in FRONT or AT BACK according to the case. (see page 4)





In order to have only one program for the eight cases, the axis block is drawn according to the following scheme with the indicated variables.

According to the case: J1, J3, J5 represent X1 or X2
J2, J4, J6 " Y1 or Y2
P1 P2 P3 P4 " X1, X2, Y1 or Y2 and are printed.
Z8 represents the minimum of the function and is printed.
Z9 " maximum " " " "
B$ and C$ represent X or Y

TITLE

General Function Z=F(X,Y) Plot

If you imagine the surface in a box, you will note that the lowest point on the screen will always be (J3,J4) with Z=Z8 and that the highest point on the screen will always be (X5,Y5) with Z=Z9.
So, when drawing the axis block, you have to watch out for the minimum M3 and the maximum M4 of the function to be eventually corrected. That is why the coordinate of the lowest point (J3,J4) when z=Z8 is reevaluated. Then with the coordinate of the opposite point in relation to the axis center (X5,Y5) with Z=Z9, the maximum M4 is reevaluated.
That allows the scales H1 and H2 to be determined correctly before drawing the axis block.
A thorough examination of the eight possible cases gives the following flowchart. (see page 11).

RECORDING OF THE DRAWING ON THE MAGNETIC TAPE.

If F ≠ 0 then the drawing is recorded in binary DATA on the tape for a quick further use. Binary data are memorized in the matrix G3( ). At the start of each line E1=0. In that case, we dimension G3 for N+1 points because we have to take into account that the edges of the surface are drawn. Varaible E is the points counter for matrix G3.
If E1≠0 then we check if the last memorized point in G3 is identical to the starting point (A1,B1) of the new segment to be drawn. Here is used the FUZZ function.

1) if YES then we memorize the arrival point (A2,B2) of the segment to be drawn in the matrix G3.

2) if NO then that means that hidden lines have been met. So we record the points already written in G3 then we resume to fill G3 with the last point.

Moreover, each time a line is finished, we have to record the matrix G3 (if F≠0)

In short we have the following flowchart (see page 12)

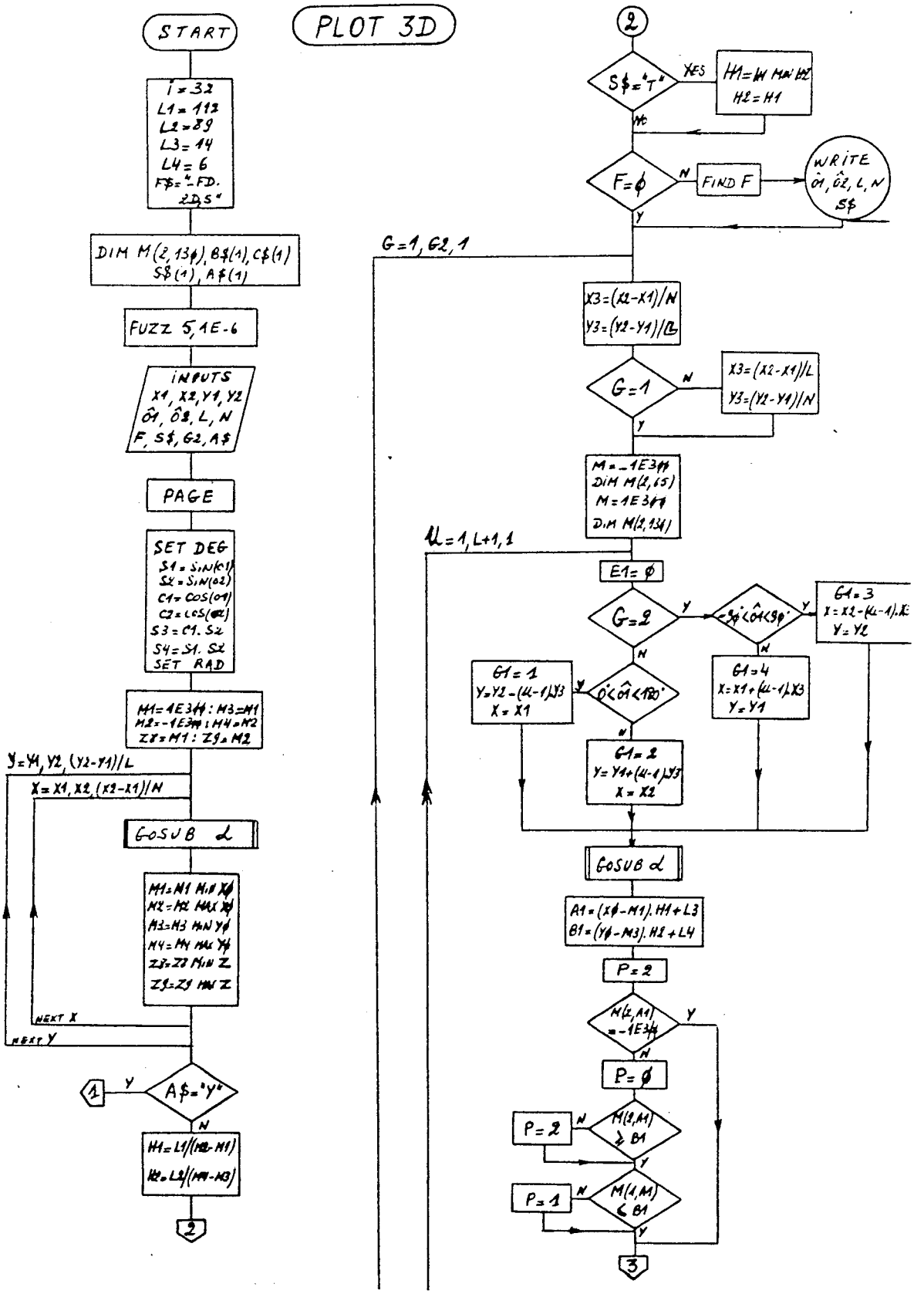
FASTER REDRAW OF THE FUNCTION.

Make sure to use an empty tape or file when you record the data of a drawing.
To read out the data in order to obtain a quick drawing, you only have to load the program in the computer and type RUN 5000 (RETURN).
The program asks for the number device and the number file to be read.
The device is the PLOTTER (number=1) or the screen (number=32).
The data are of course read exactly as they have been recorded, i.e in GDU.
(see the flowchart on page 12).

REMARKS: 1) It is obvious that the found minimum Z8 and maximum Z9 which are printed are not necessarely the true values. The more the number of slices L and the number of points N are great, the more Z8 and Z9 will be close to reality.
2) The axis block can also be recorded on tape. Personally I have not had enough memory to do so.

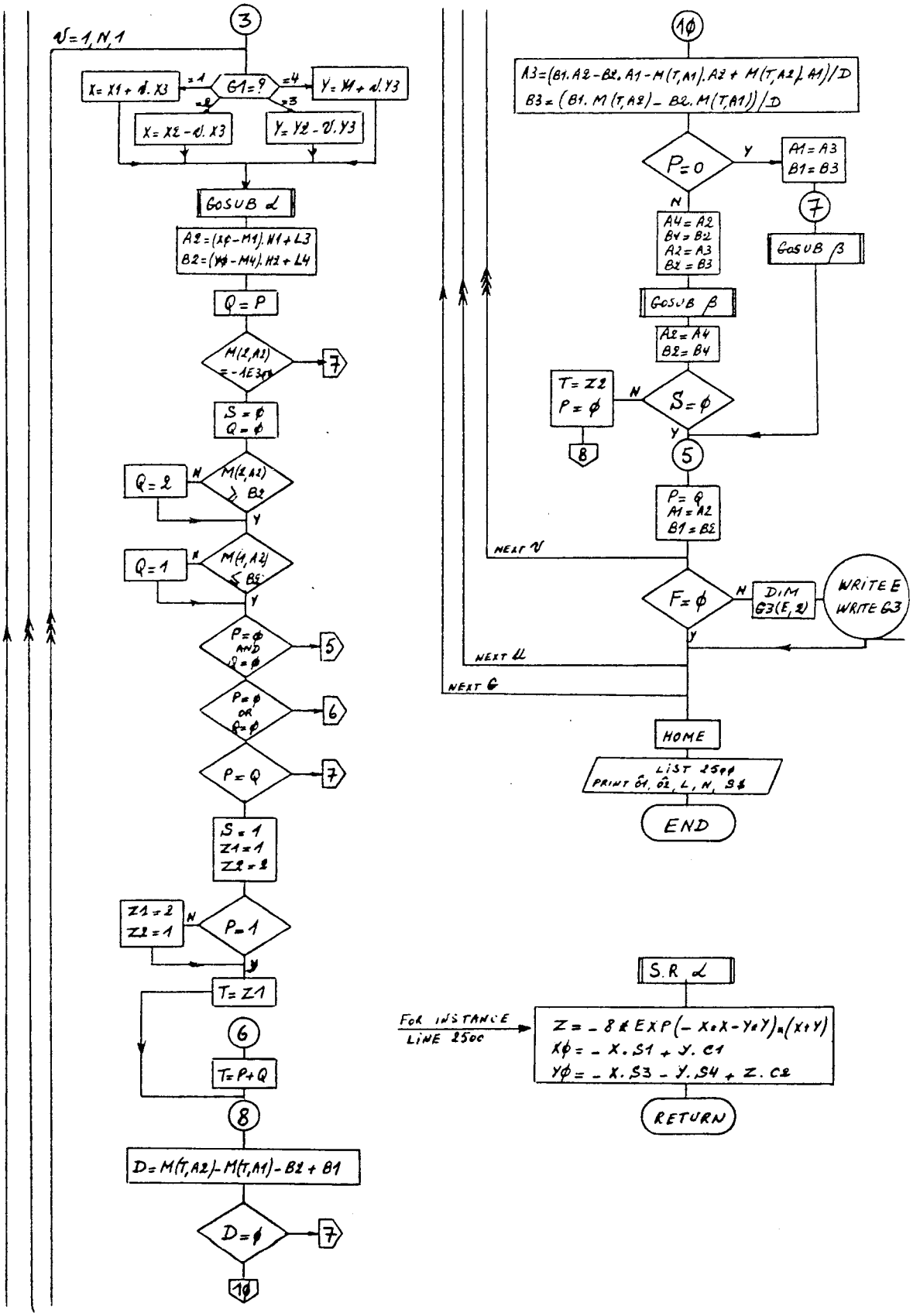
TITLE

General Function Z=F(X,Y) Plot



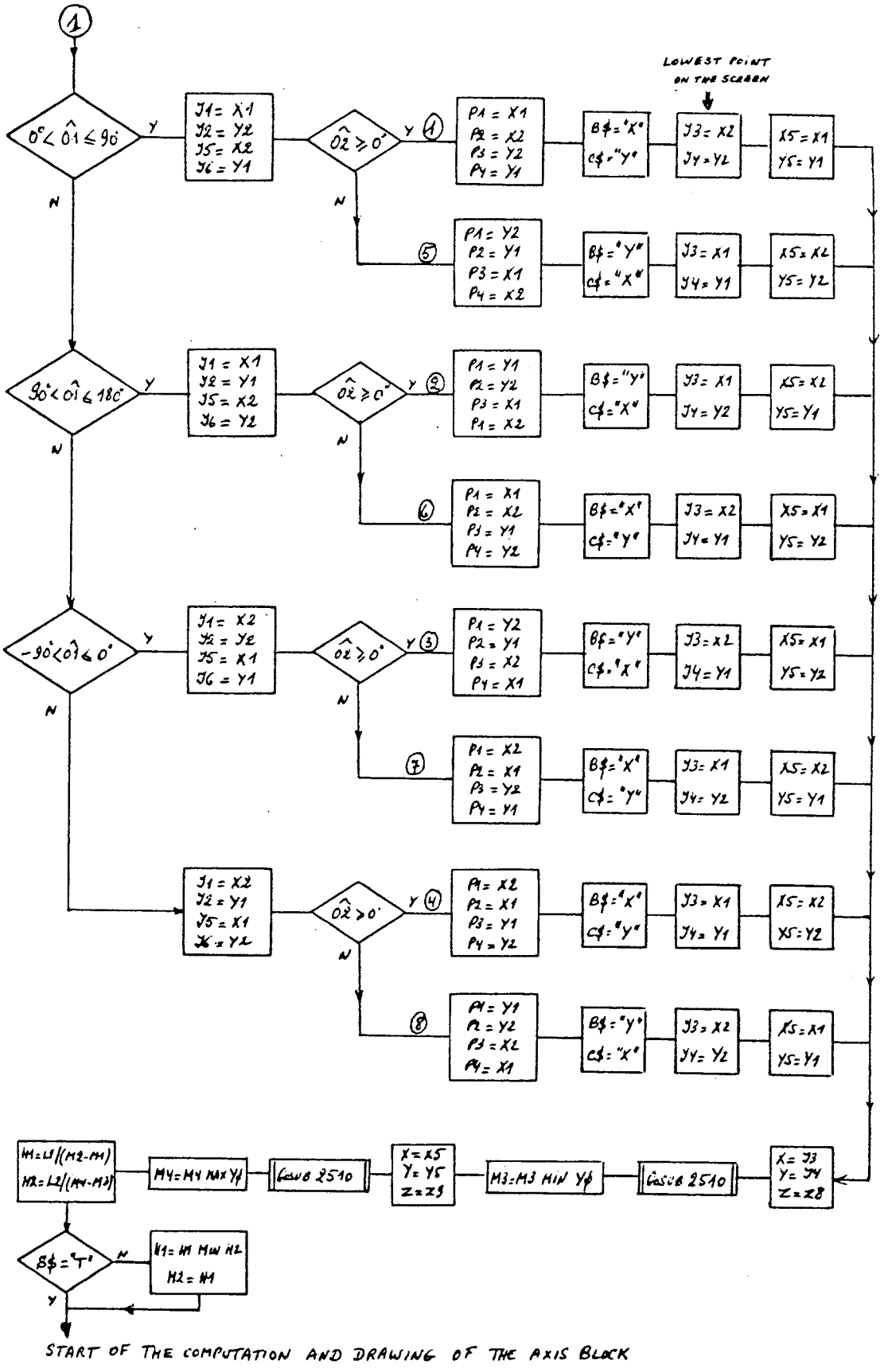
TITLE

General Function Z=F(X,Y) Plot



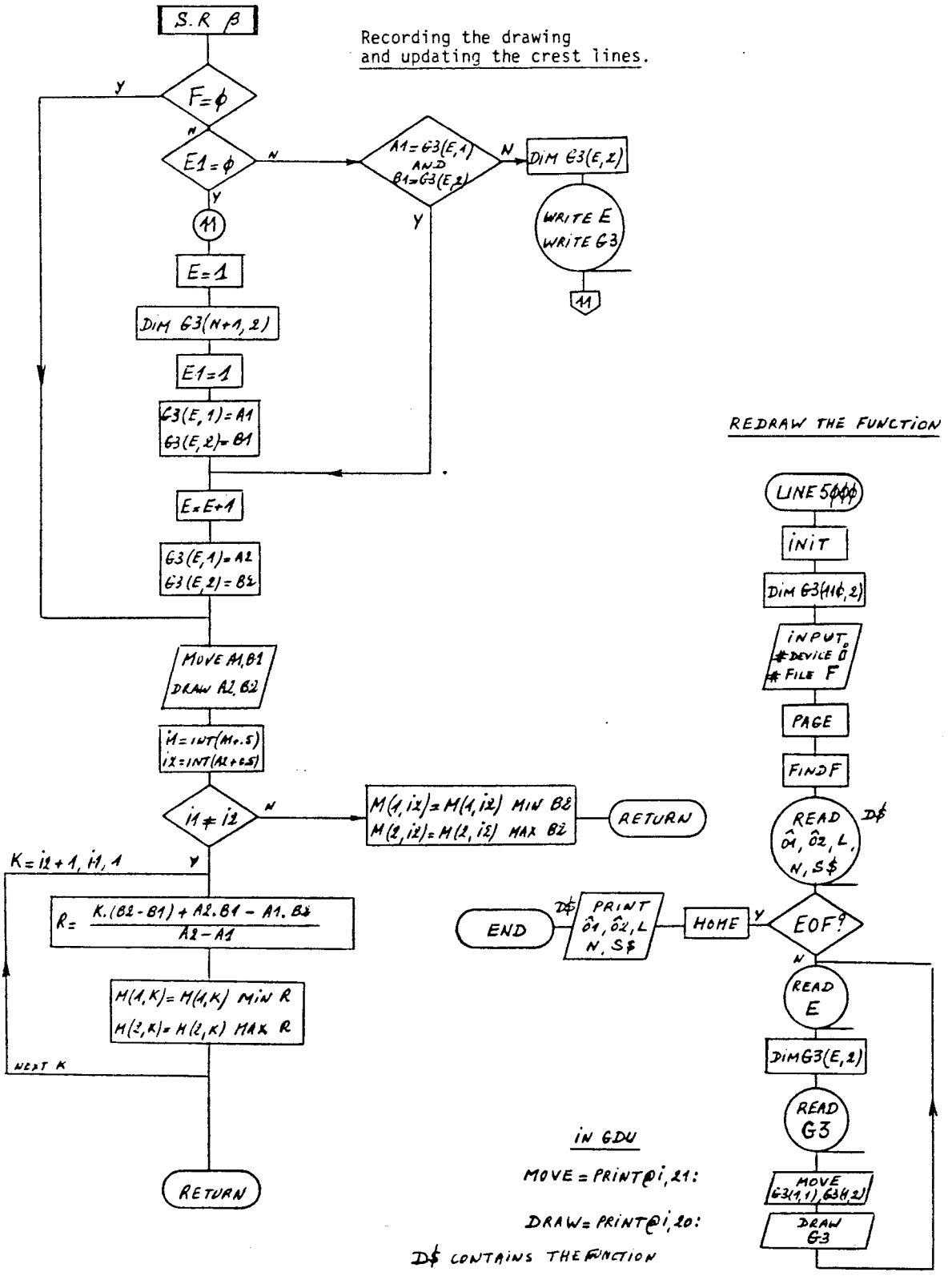
TITLE

General Function Z=F(X,Y) Plot



TITLE

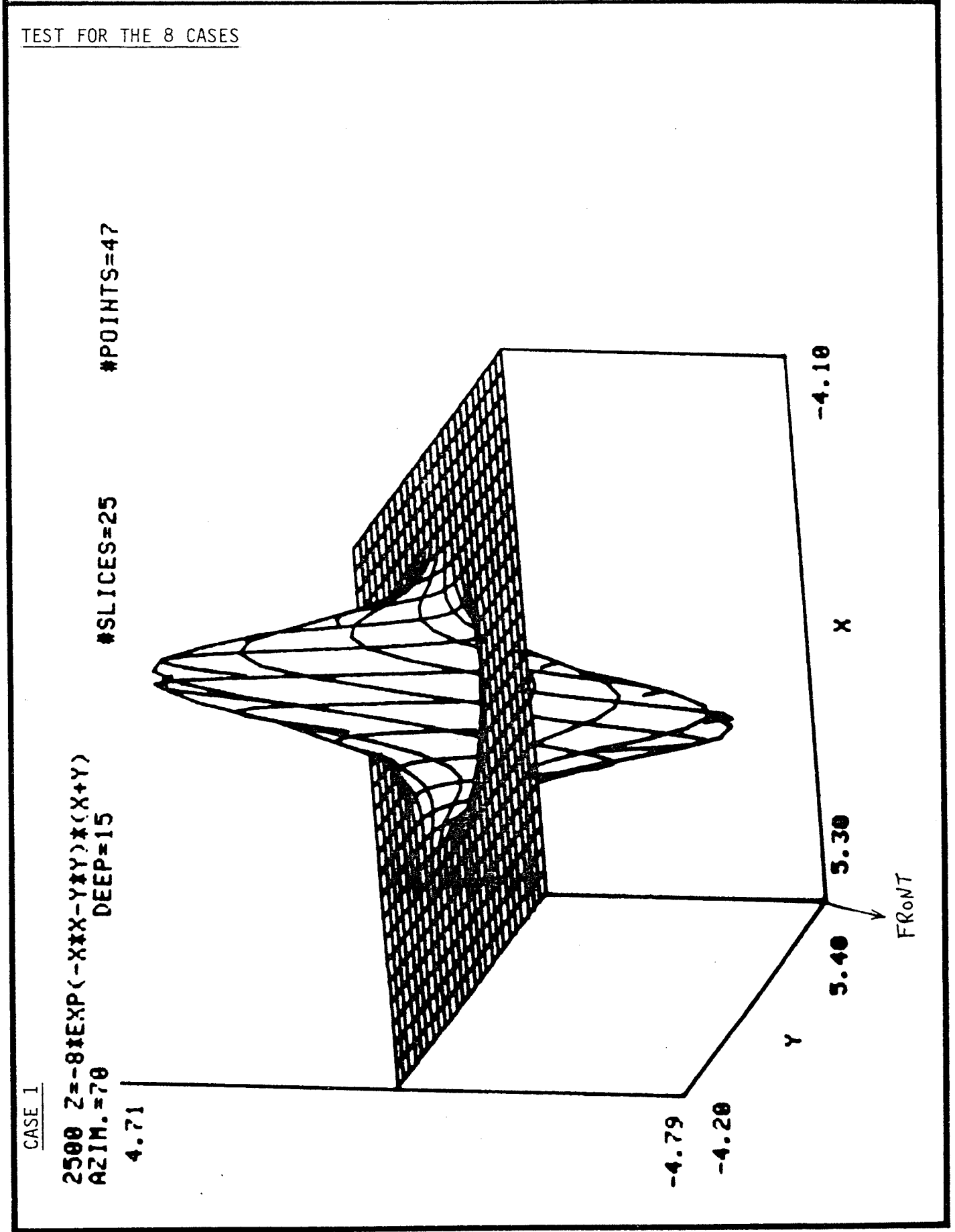
General Function Z=F(X,Y) Plot



TITLE

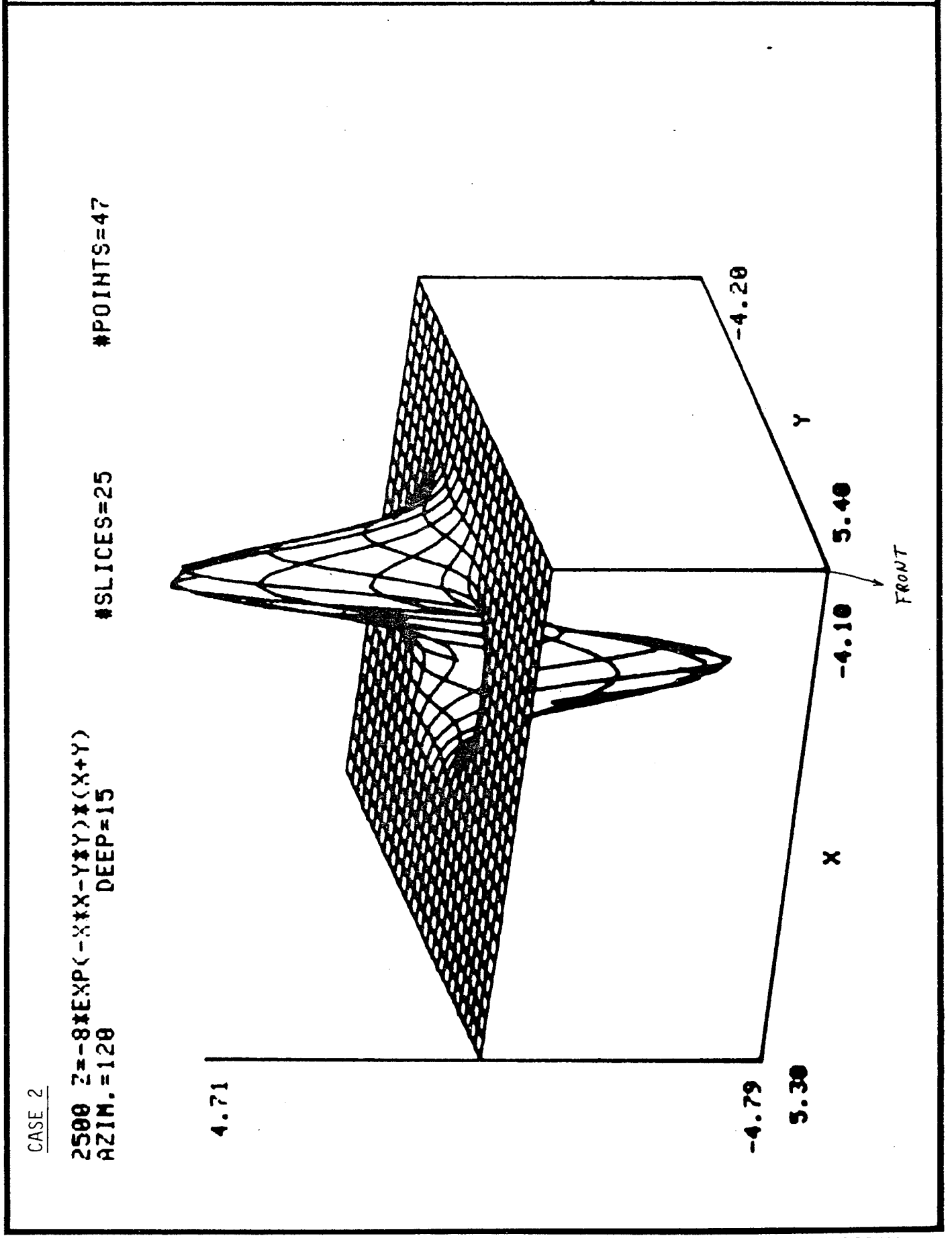
General Function Z=F(X,Y) Plot

TEST FOR THE 8 CASES



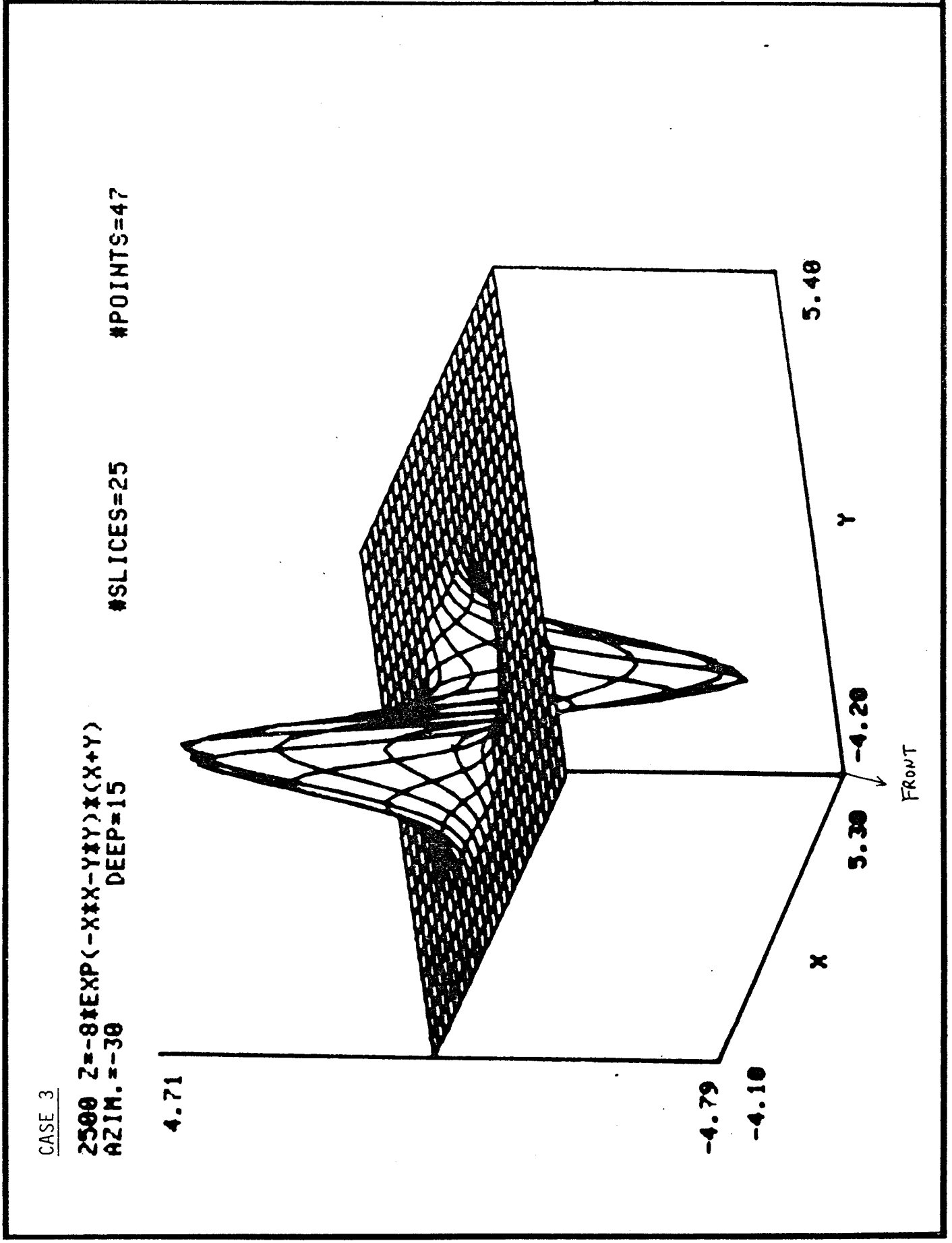
TITLE

General Function Z=F(X,Y) Plot



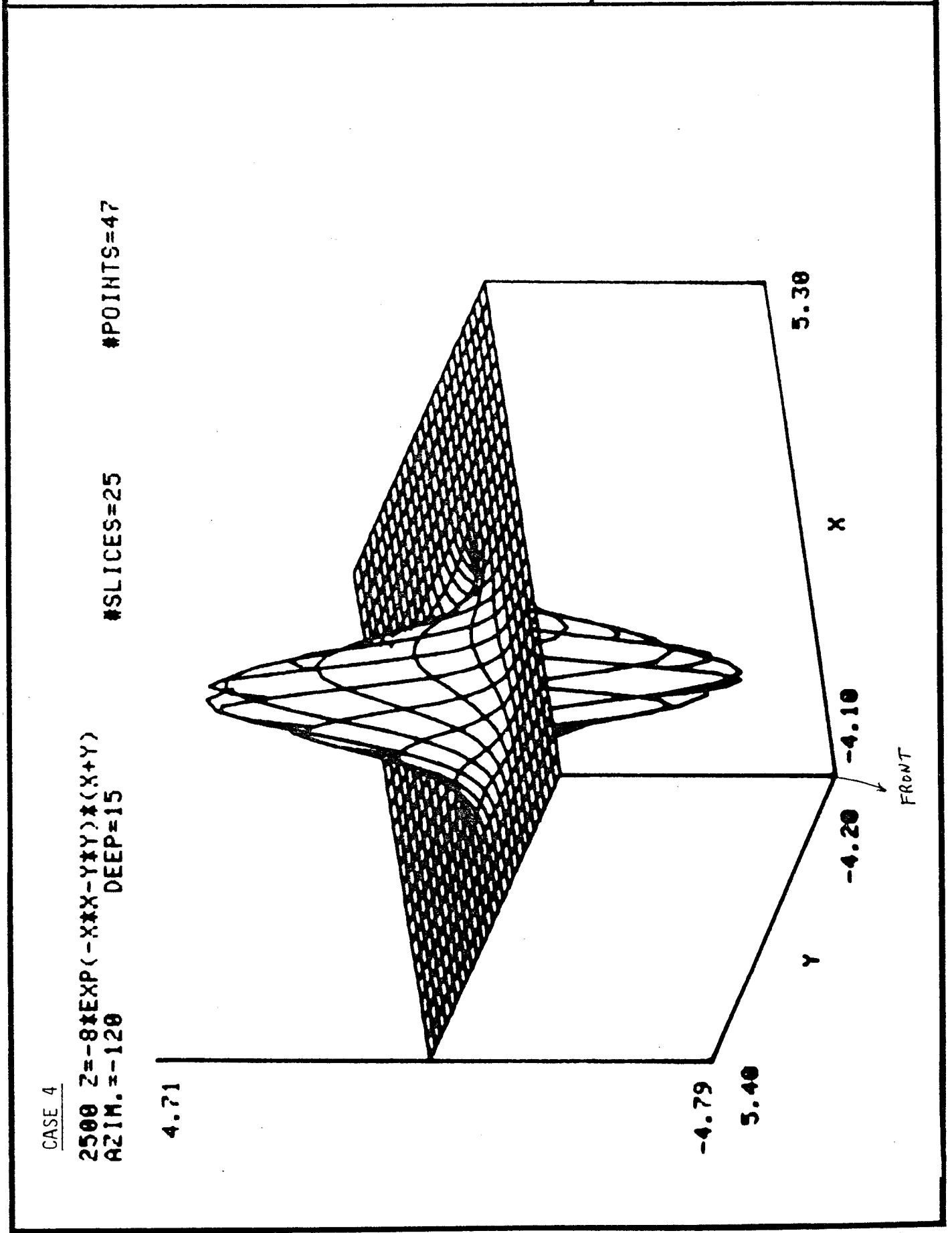
TITLE

General Function Z=F(X,Y) Plot



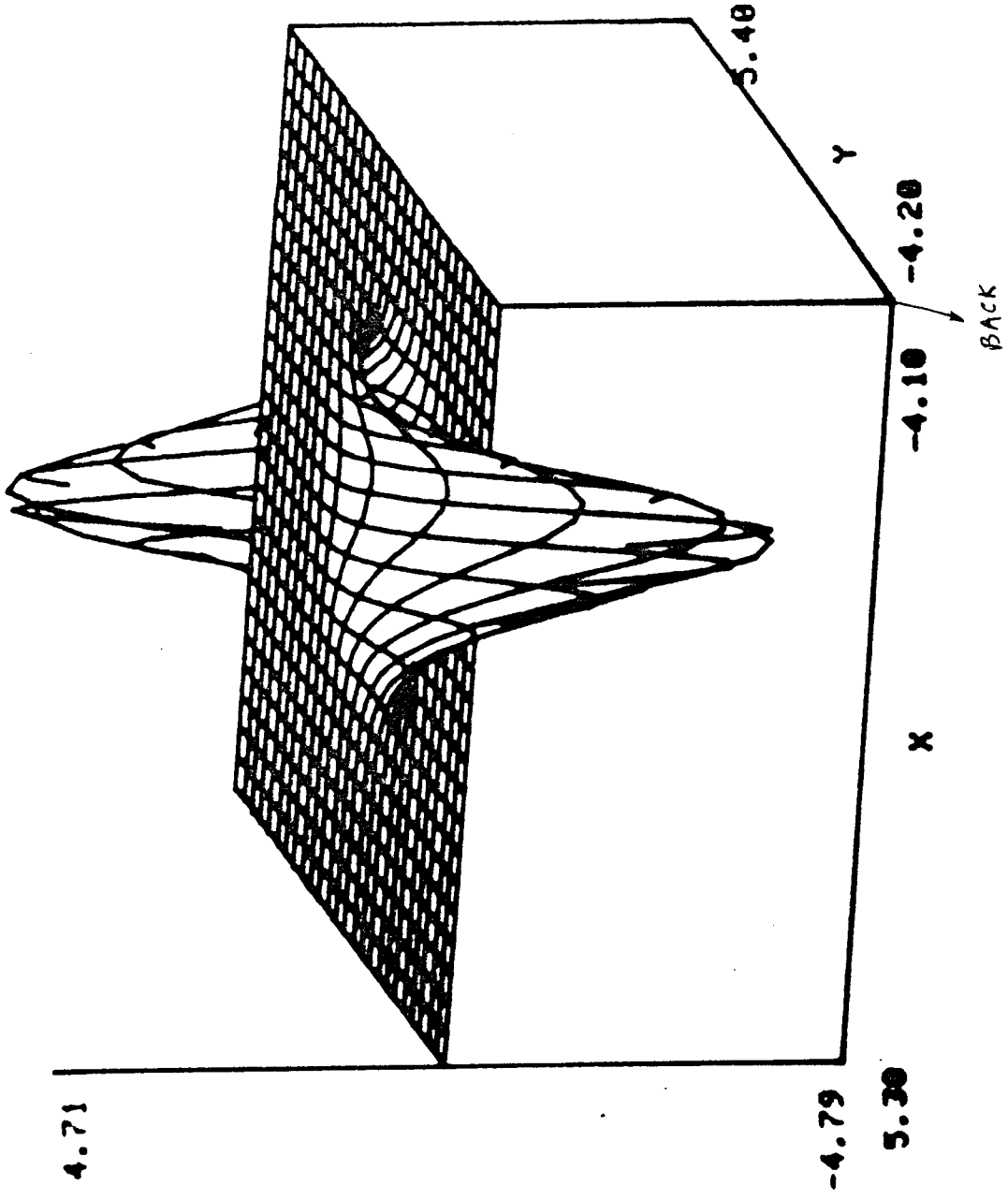
TITLE

General Function Z=F(X,Y) Plot



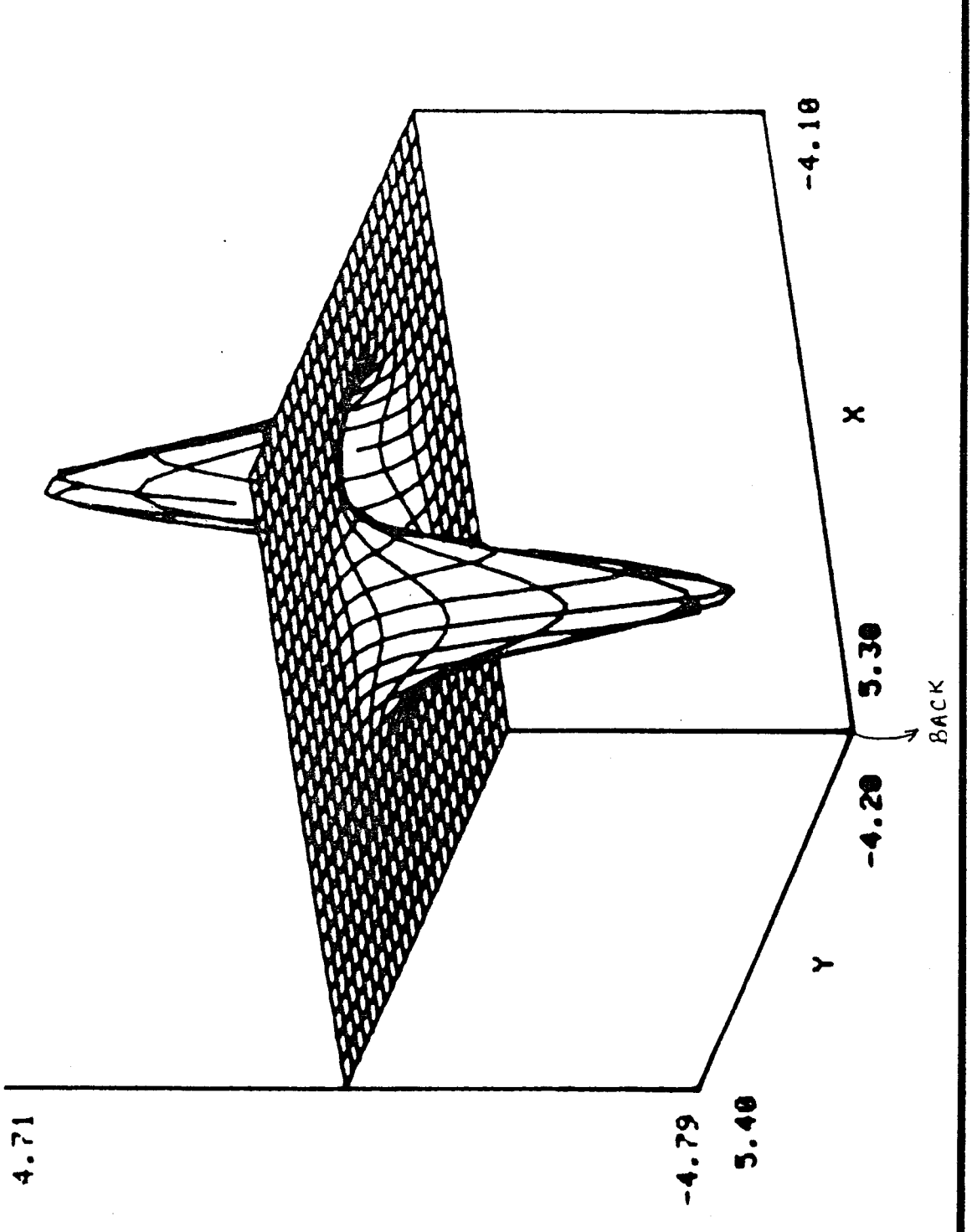
TITLE

General Function Z=F(X,Y) Plot



TITLE

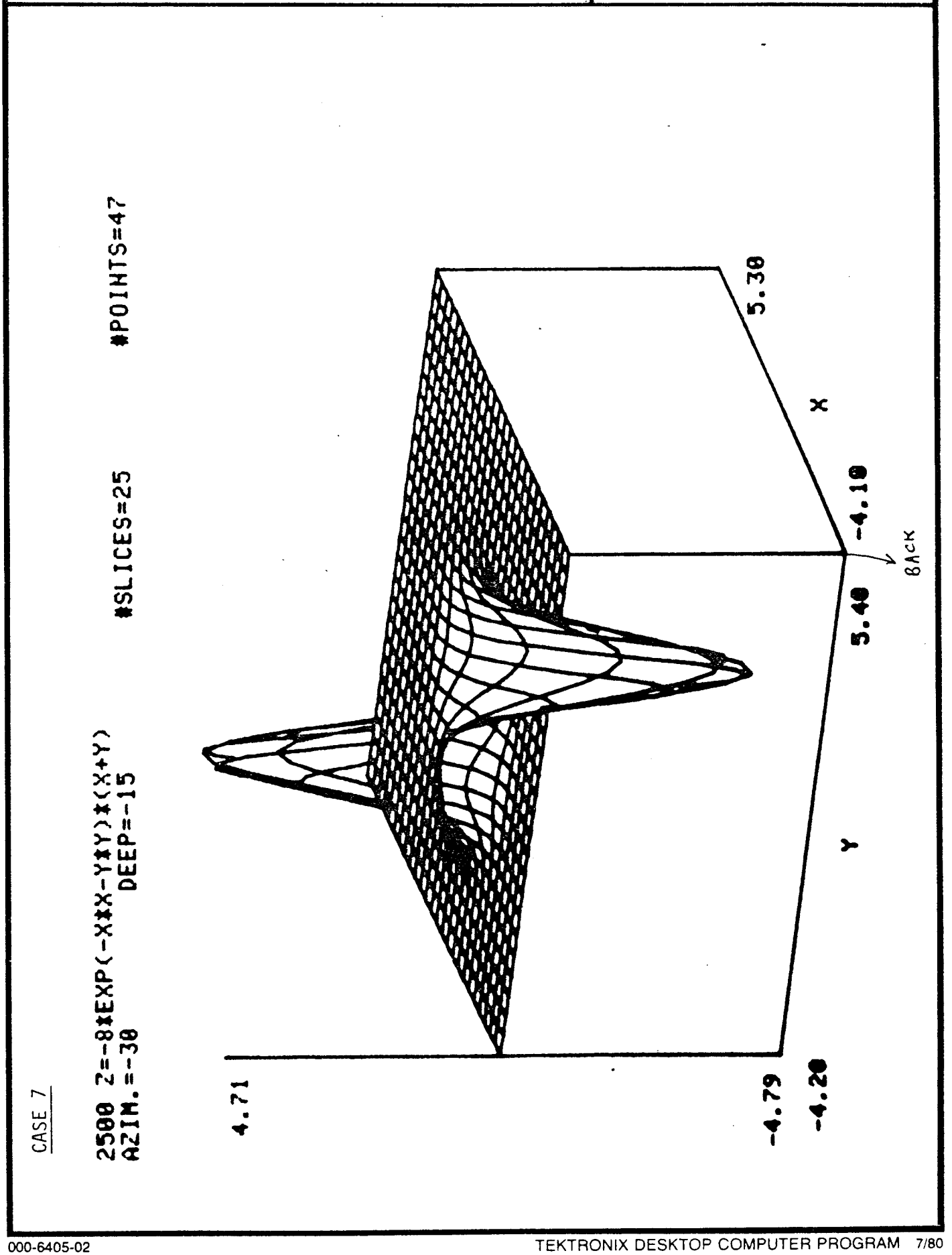
General Function Z=F(X,Y) Plot



CASE 6

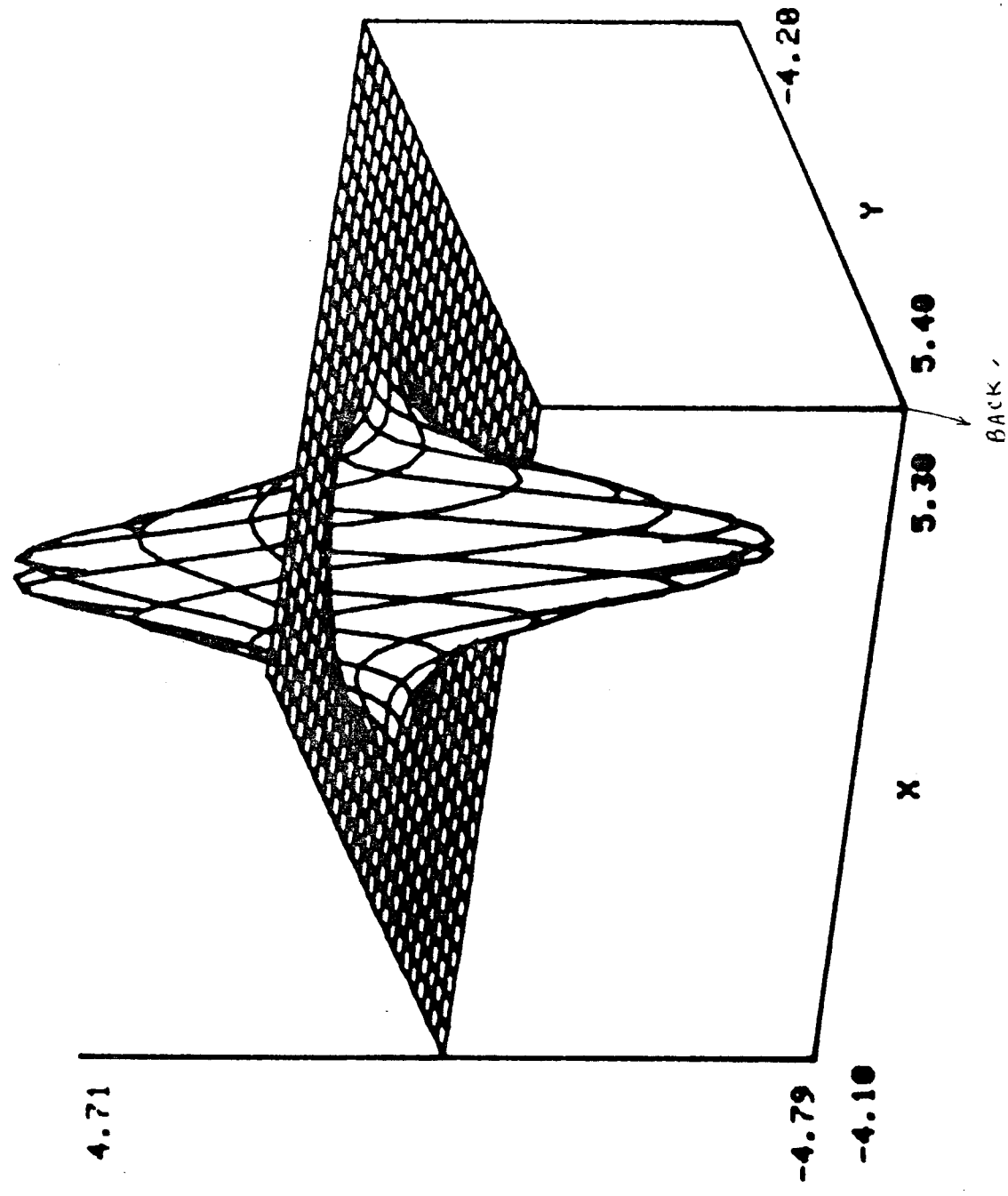
TITLE

General Function Z=F(X,Y) Plot



TITLE

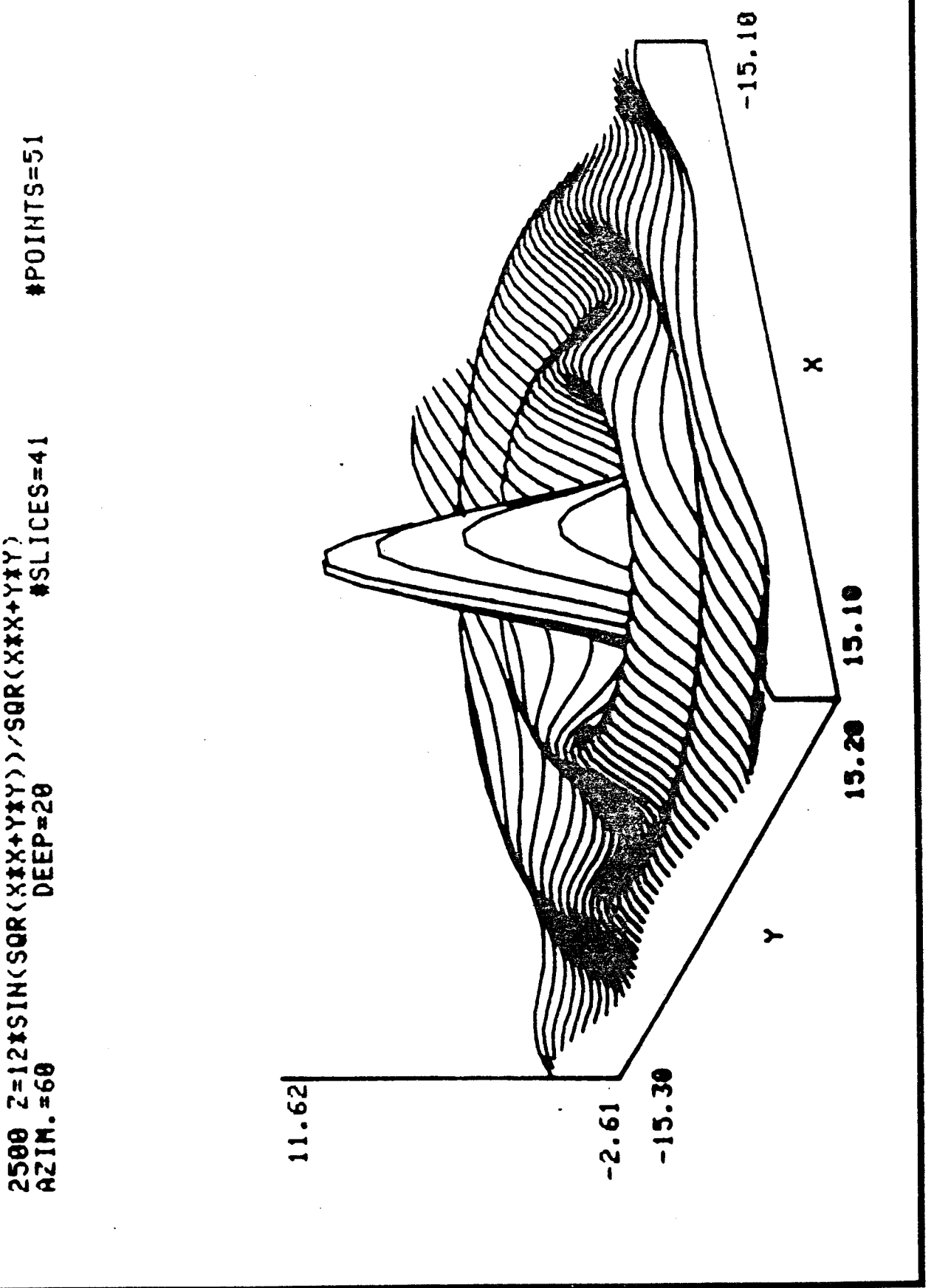
General Function Z=F(X,Y) Plot



CASE 8

TITLE

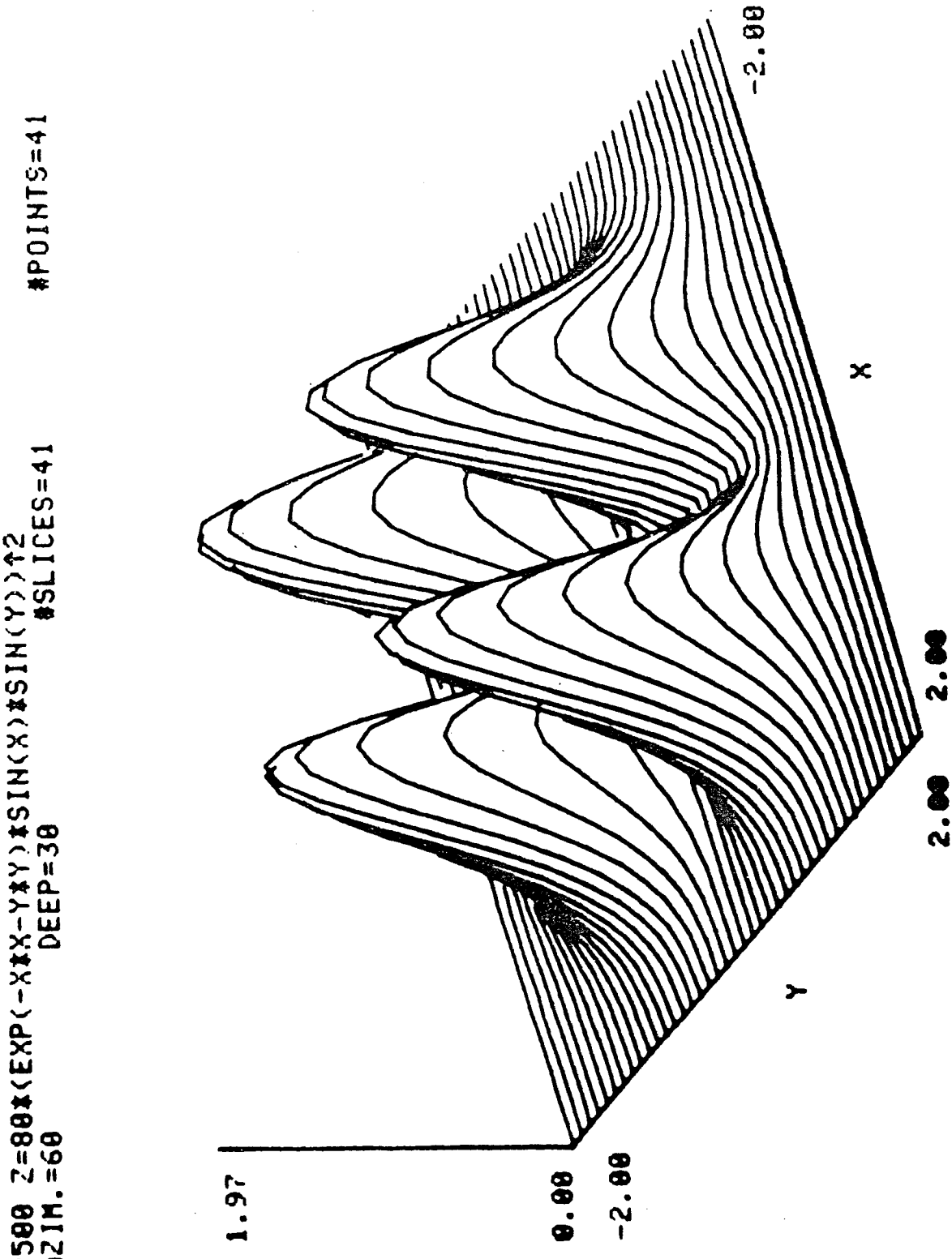
General Function Z=F(X,Y) Plot



TITLE

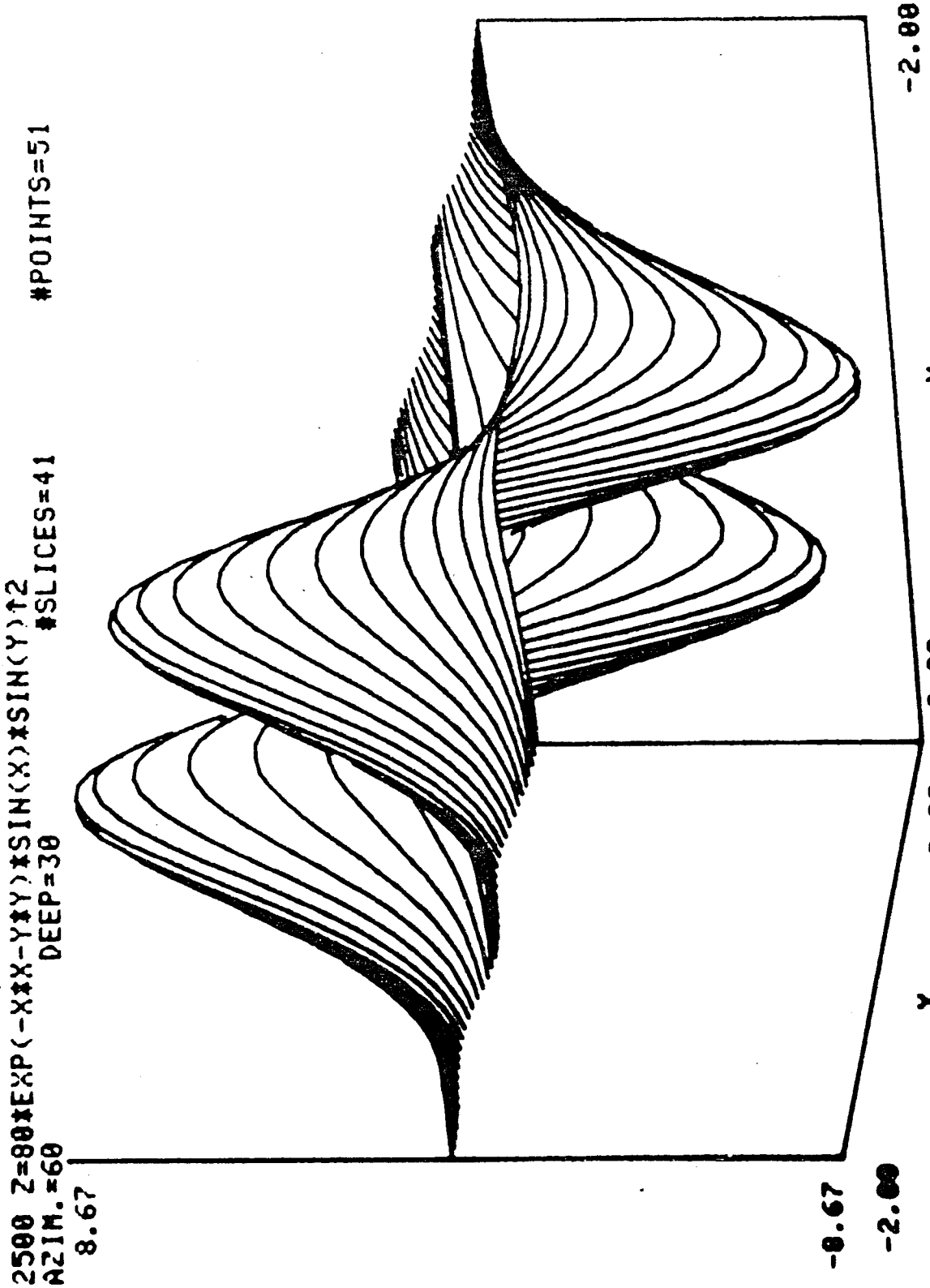
General Function Z=F(X,Y) Plot

EXAMPLES



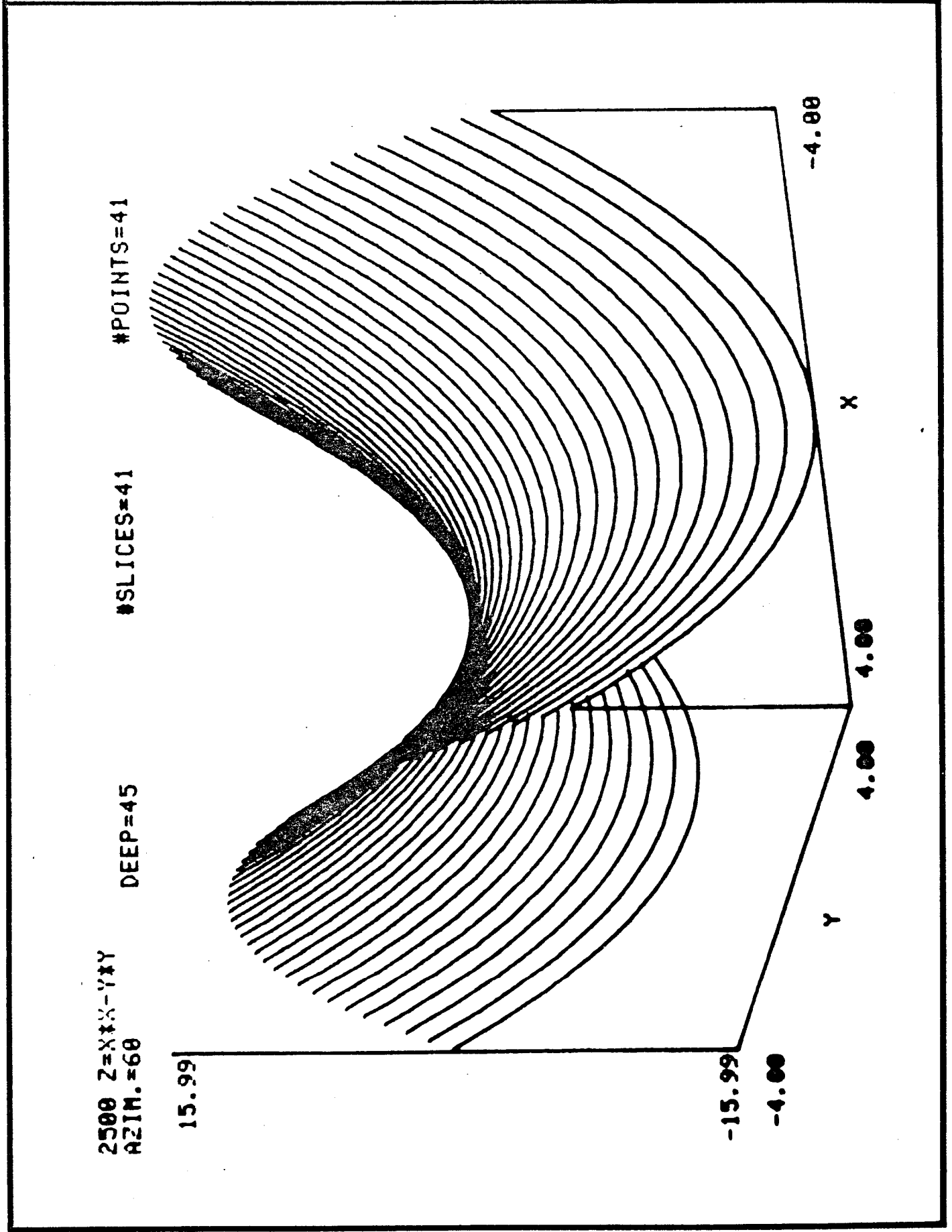
TITLE

General Function Z=F(X,Y) Plot



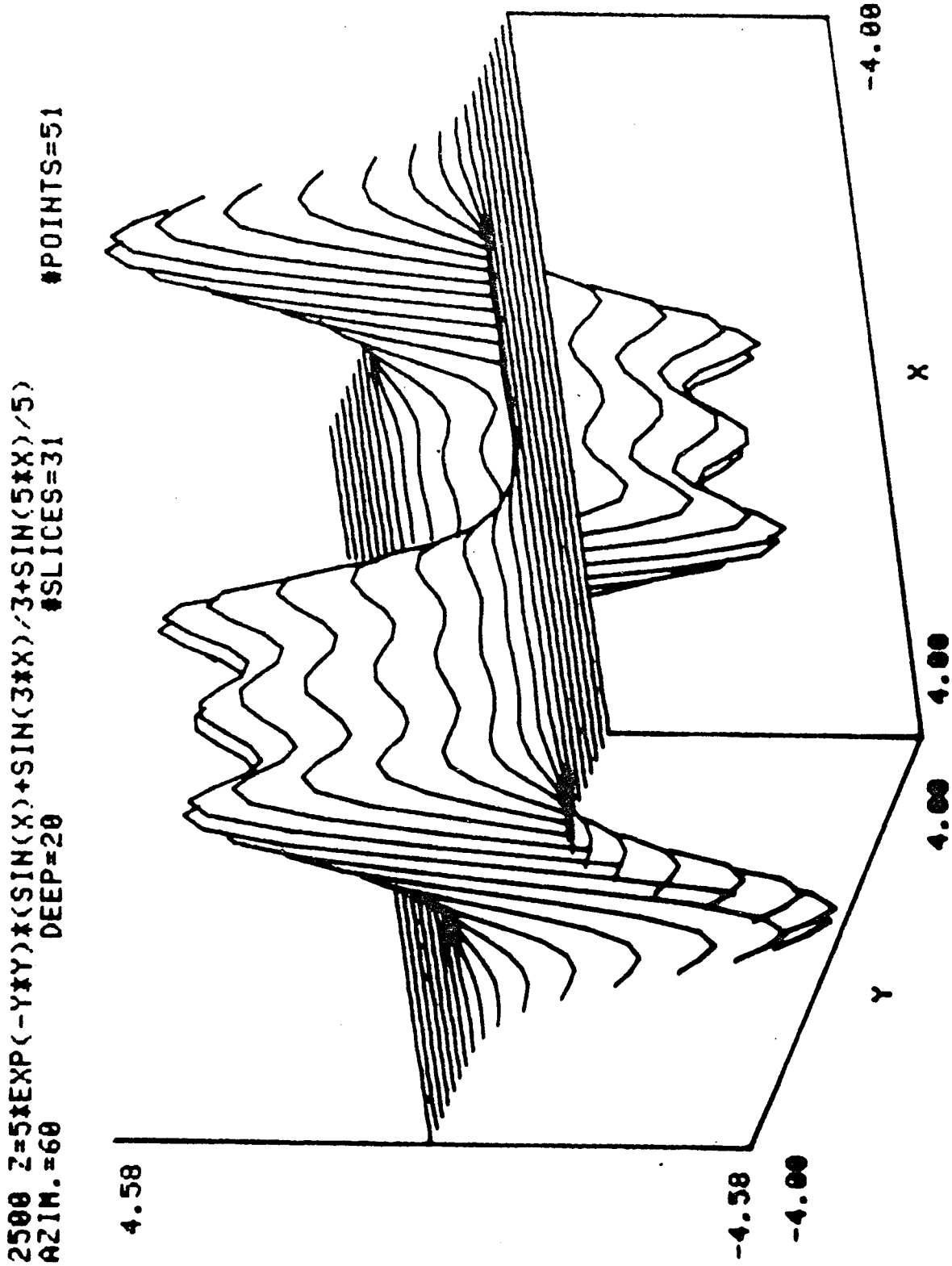
TITLE

General Function Z=F(X,Y) Plot



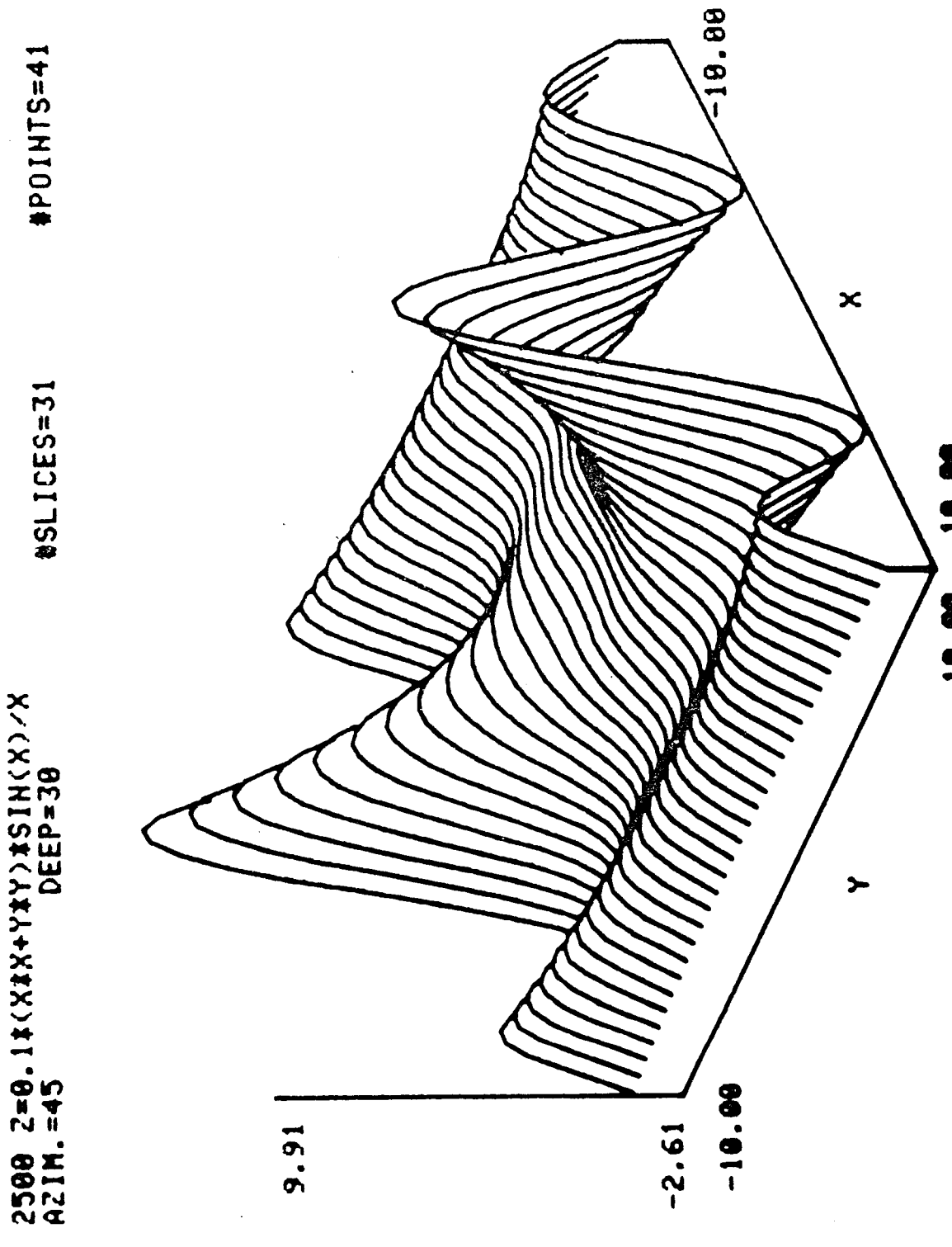
TITLE

General Function Z=F(X,Y) Plot



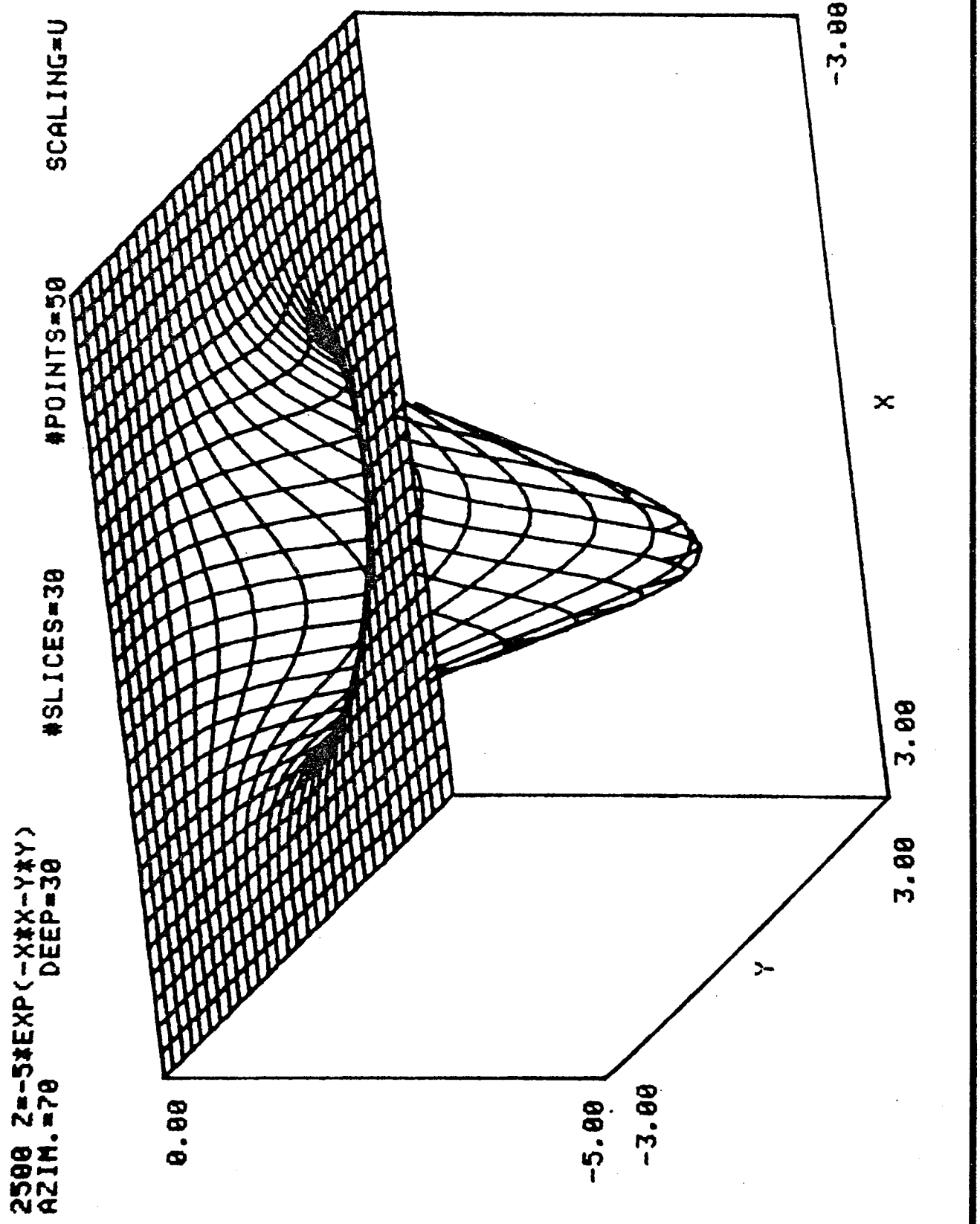
TITLE

General Function Z=F(X,Y) Plot



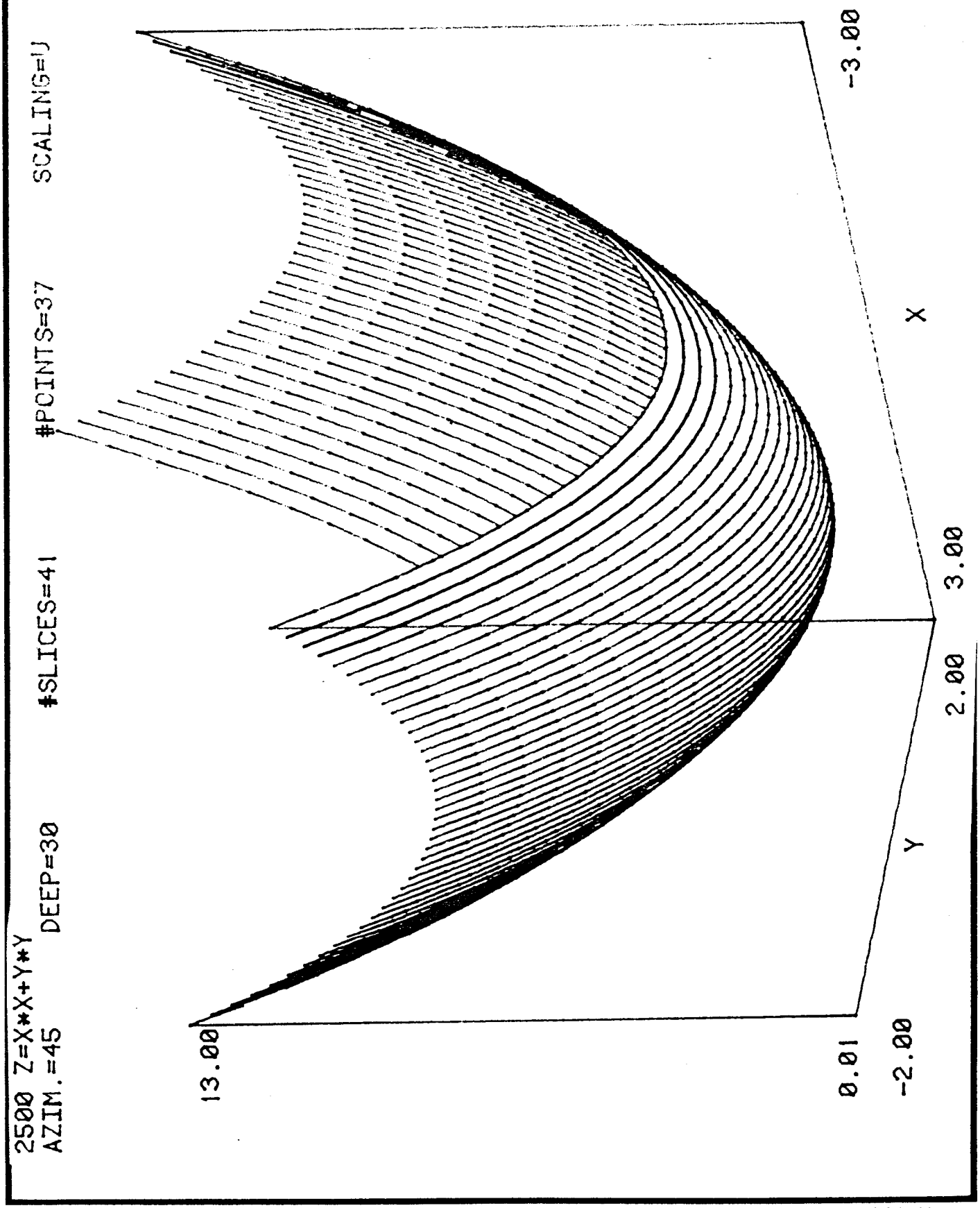
TITLE

General Function Z=F(X,Y) Plot



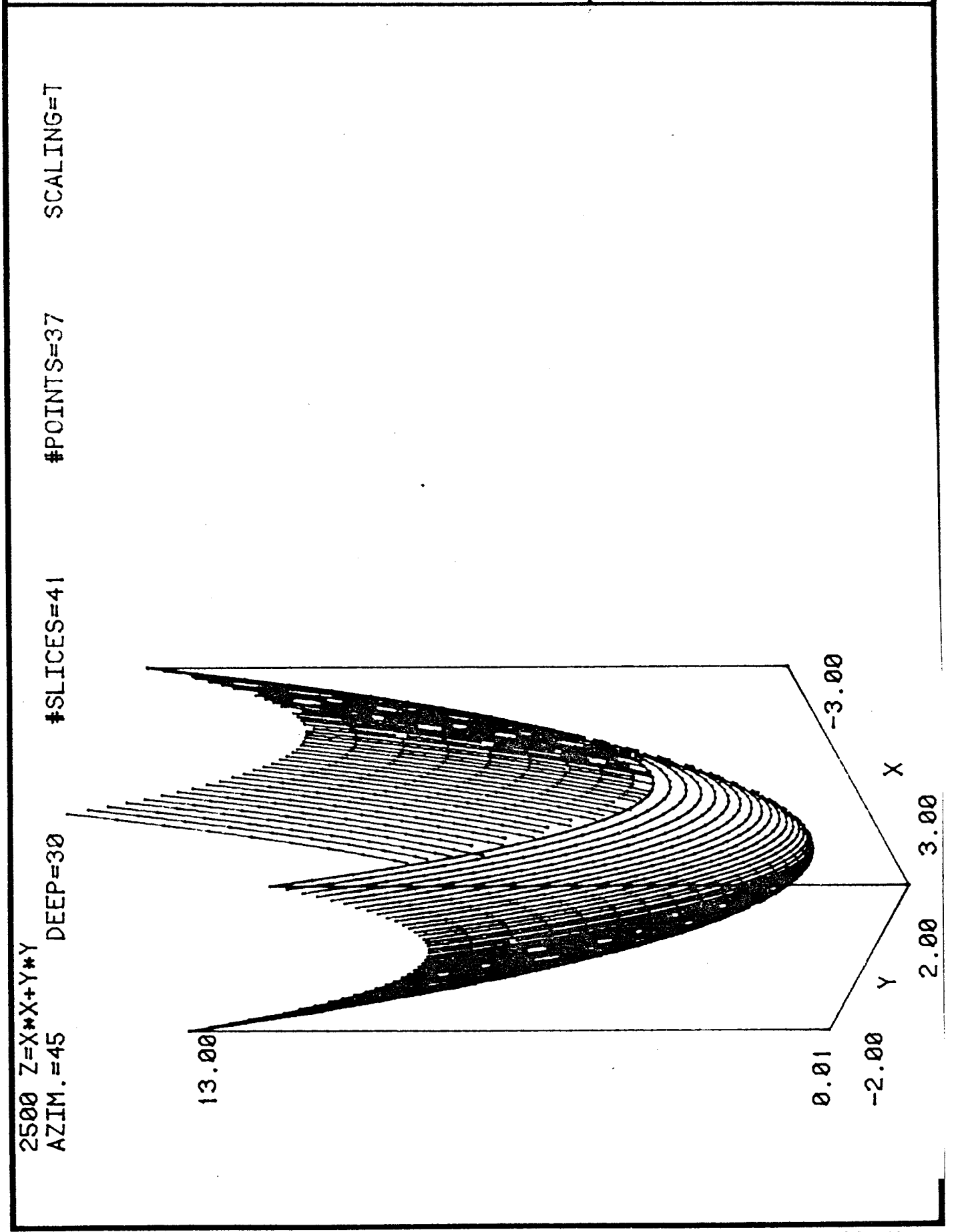
TITLE

General Function Z=F(X,Y) Plot



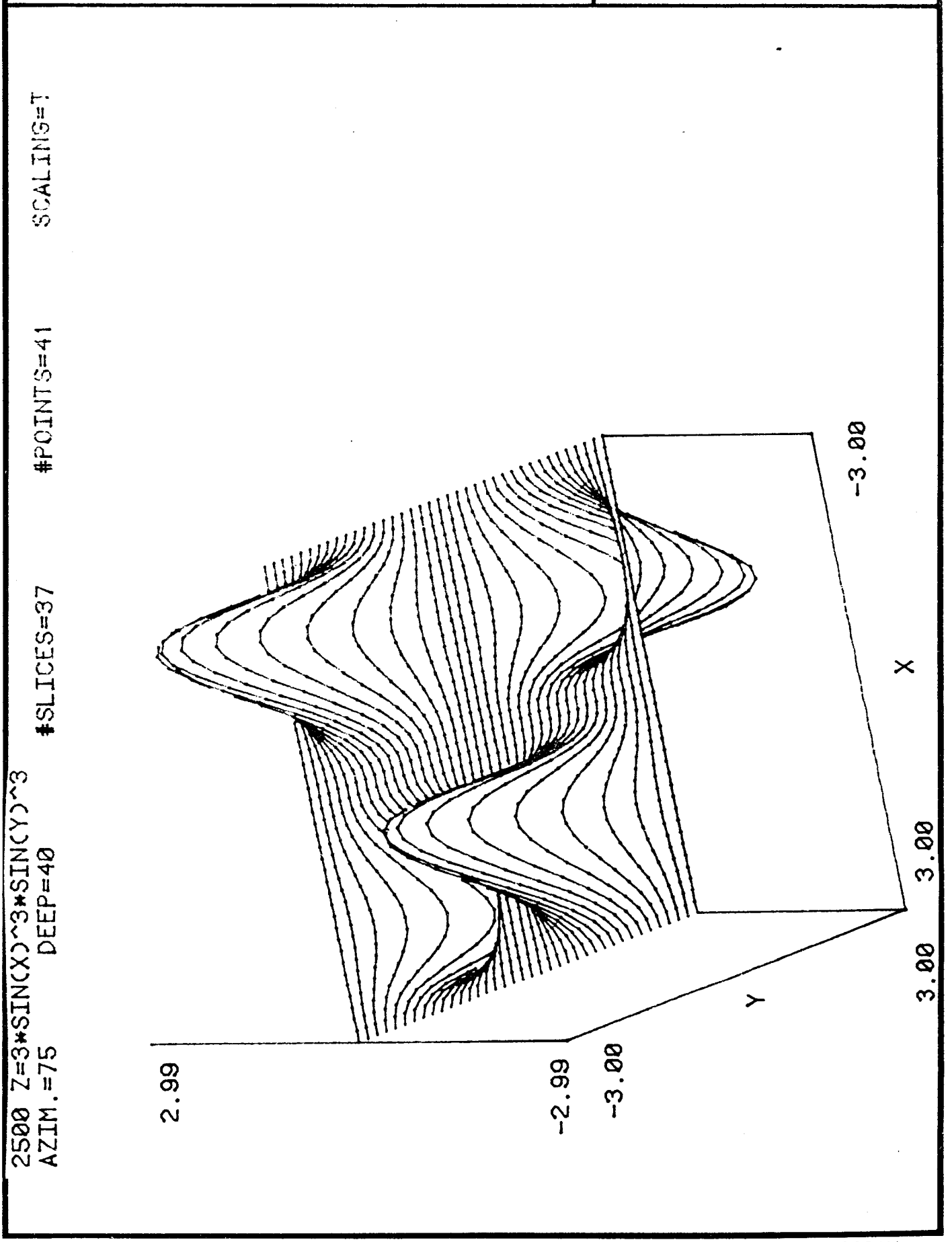
TITLE

General Function Z=F(X,Y) Plot



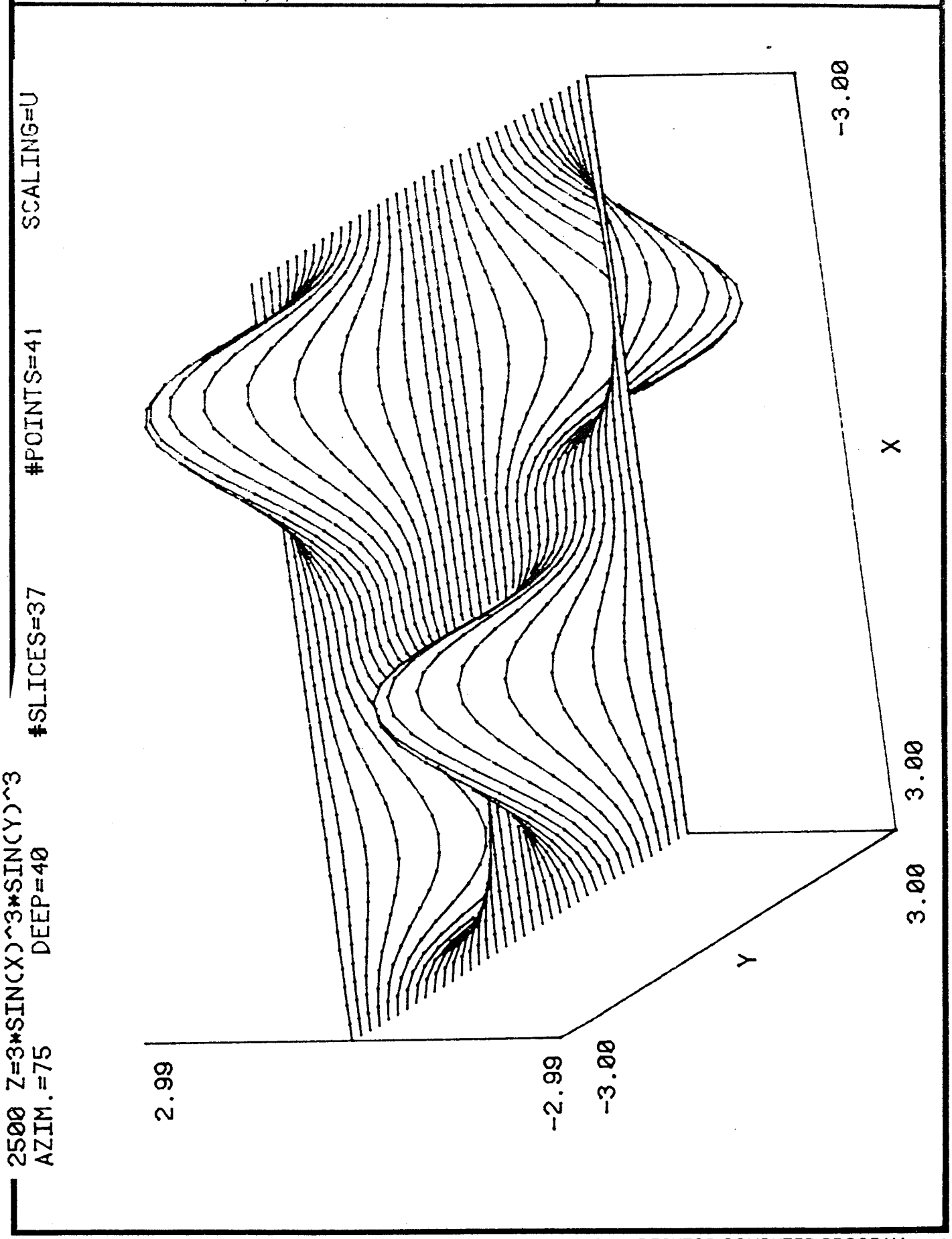
TITLE

General Function Z=F(X,Y) Plot



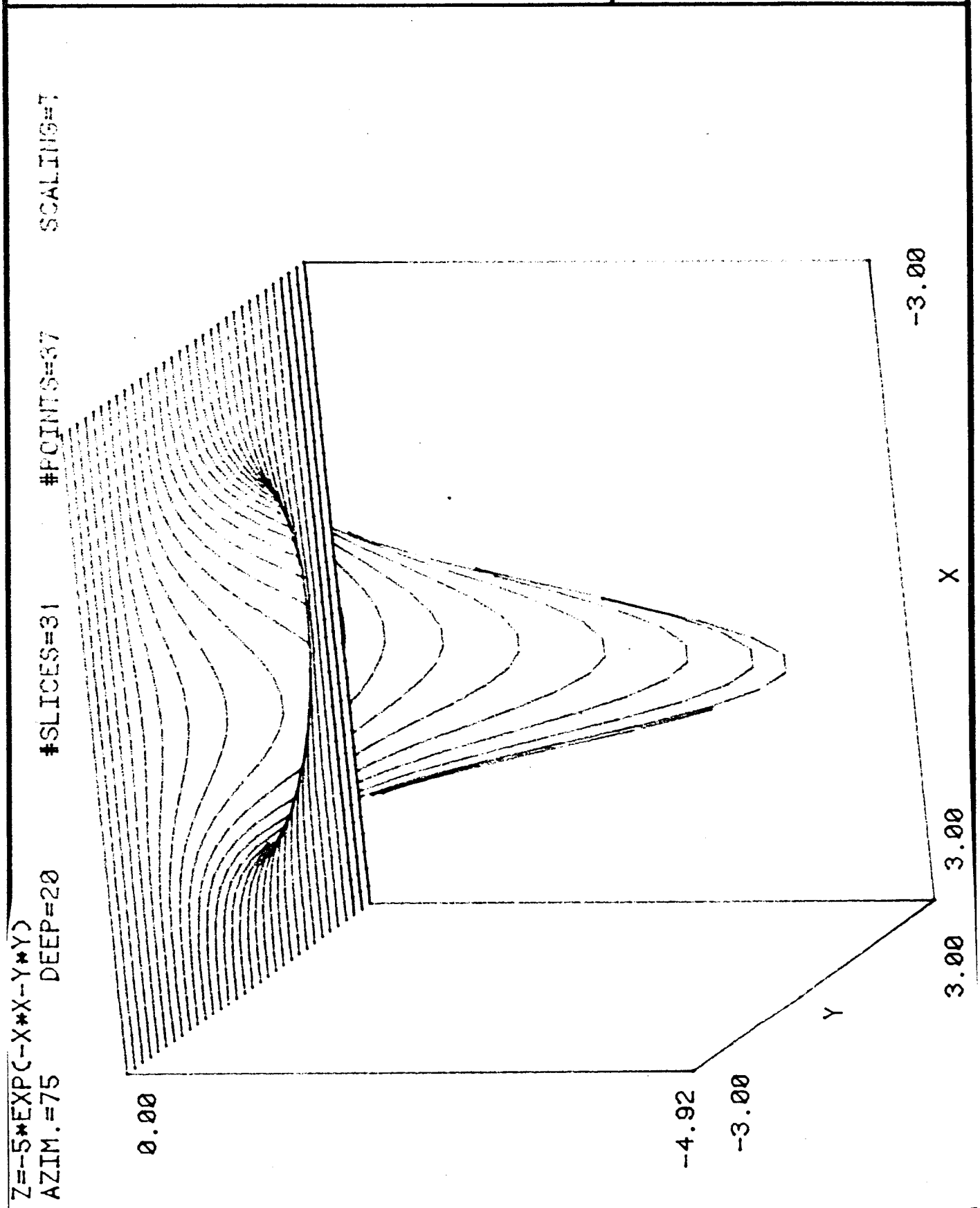
TITLE

General Function Z=F(X,Y) Plot



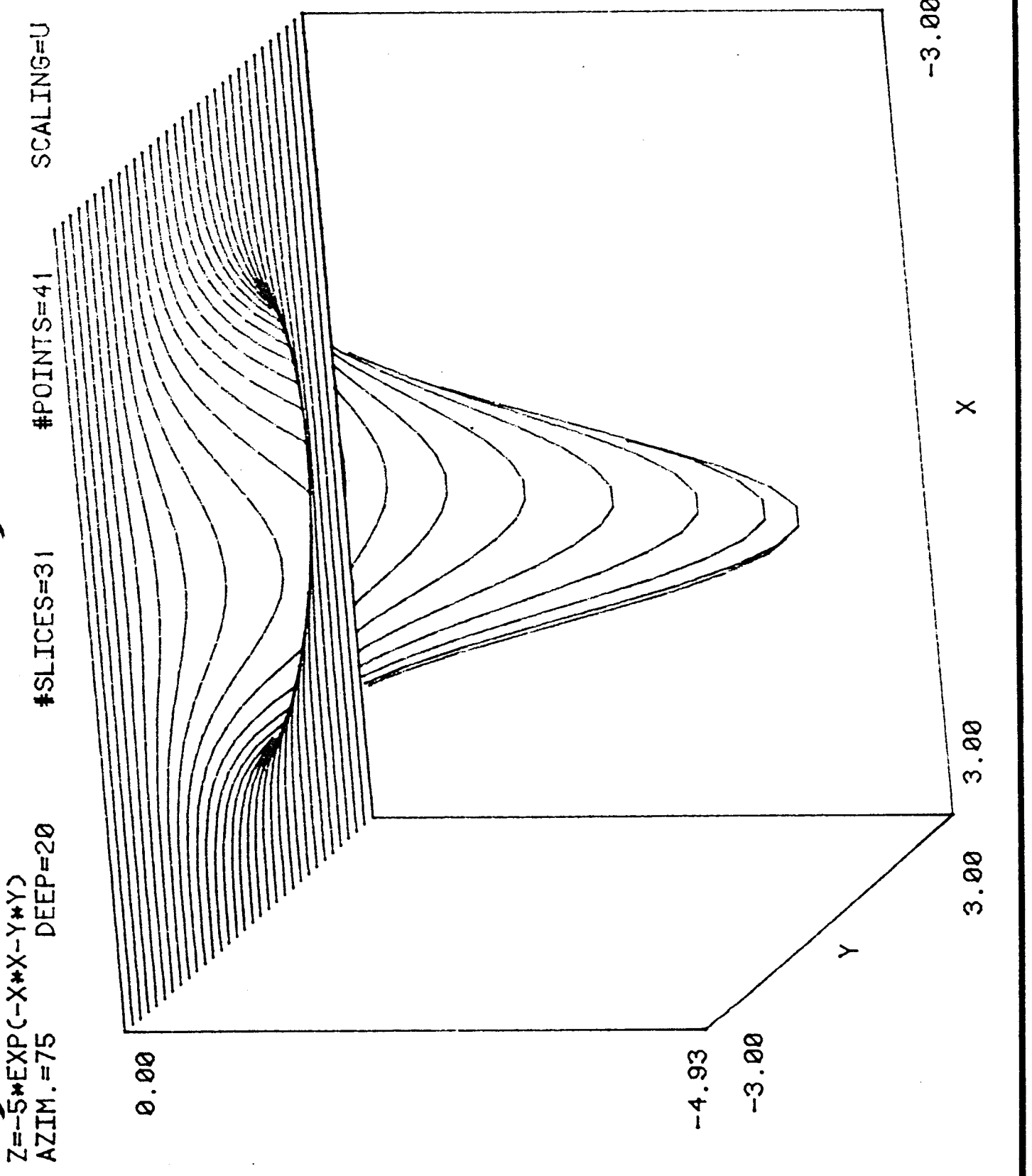
TITLE

General Function Z=F(X,Y) Plot



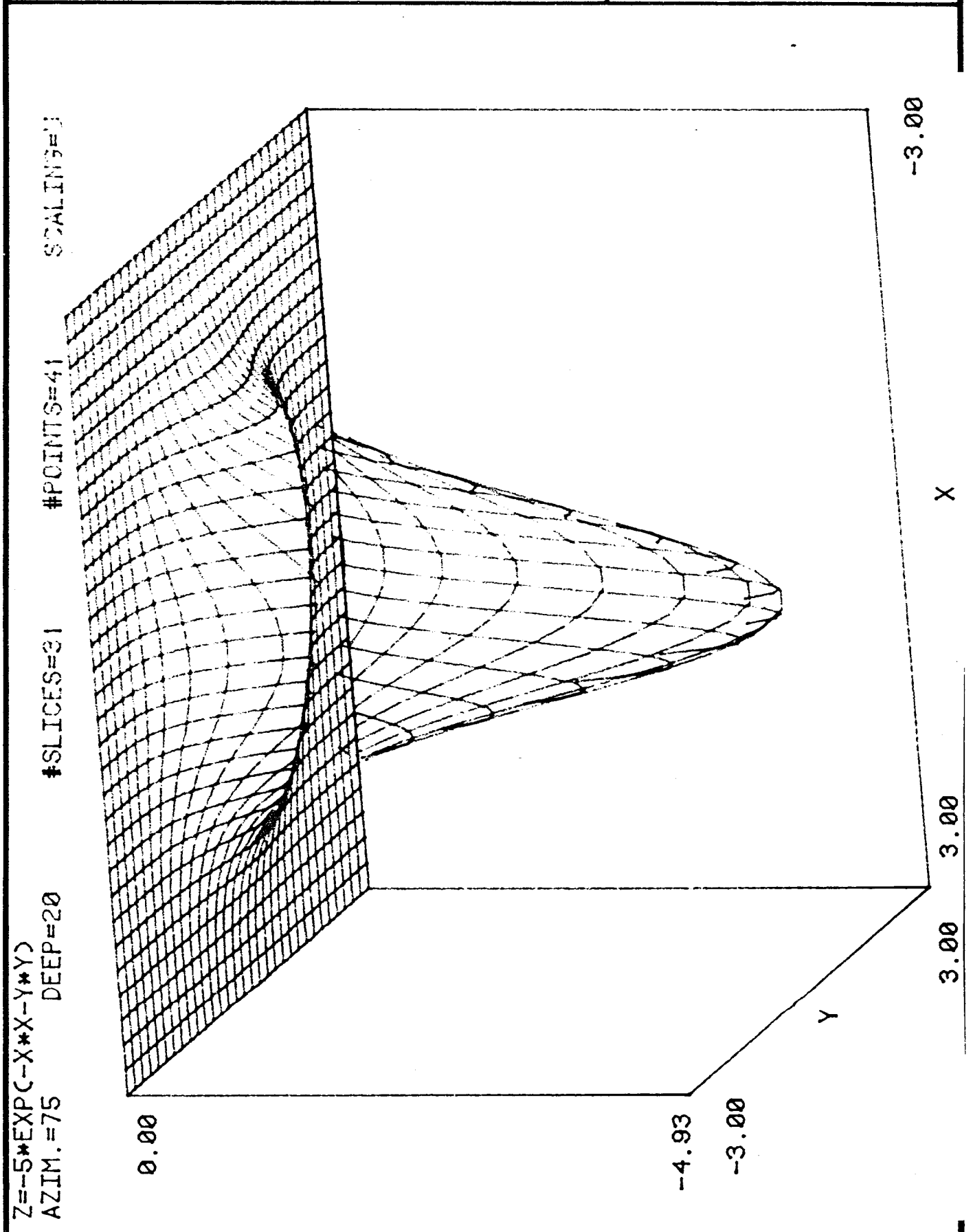
TITLE

General Function Z=F(X,Y) Plot



TITLE

General Function Z=F(X,Y) Plot



Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
Stereo Surface		
ORIGINAL DATE	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
March, 1981		4052/4
AUTHOR	University of Nottingham	PERIPHERALS
Dr. P. R. Tregenza	Nottingham, England	Optional-4662/4663 Plotter

ABSTRACT

Files: 1 ASCII Program

Statements: 200

An equation of the form Z=F(X,Y) is plotted as a surface drawn in perspective with optional plotter output as a stereo pair. The equation is contained in a subroutine which is listed initially on the screen. The user may change the perspective viewpoint and specify the number of lines drawn. Hidden lines are removed from the image.

Functions must be continuous and injective.

TITLE

Stereo Surface

OPERATING INSTRUCTIONS

Load the program into memory throug h the tape directory or FIND 34 and OLD.

After the initial RUN command, the screen displays the subroutine containing the function to be plotted. To change the function, press BREAK key twice and recode lines 3010 to 3999 with your function. Then type RUN again followed by pressing RETURN key.

When the screen displays the subroutine containing the function to be plotted, press RETURN to allow the program to continue.

You will be prompted for the following input:

X and Y minimum and maximum

Output Format - Enter 1 for single drawing on plotter
" 2 for stereo pair
or just press RETURN for screen drawing

Number of drawn lines - This depends on drawing size, 40 - 60 being appropriate for a drawing about 15" square.

The drawing is then produced.

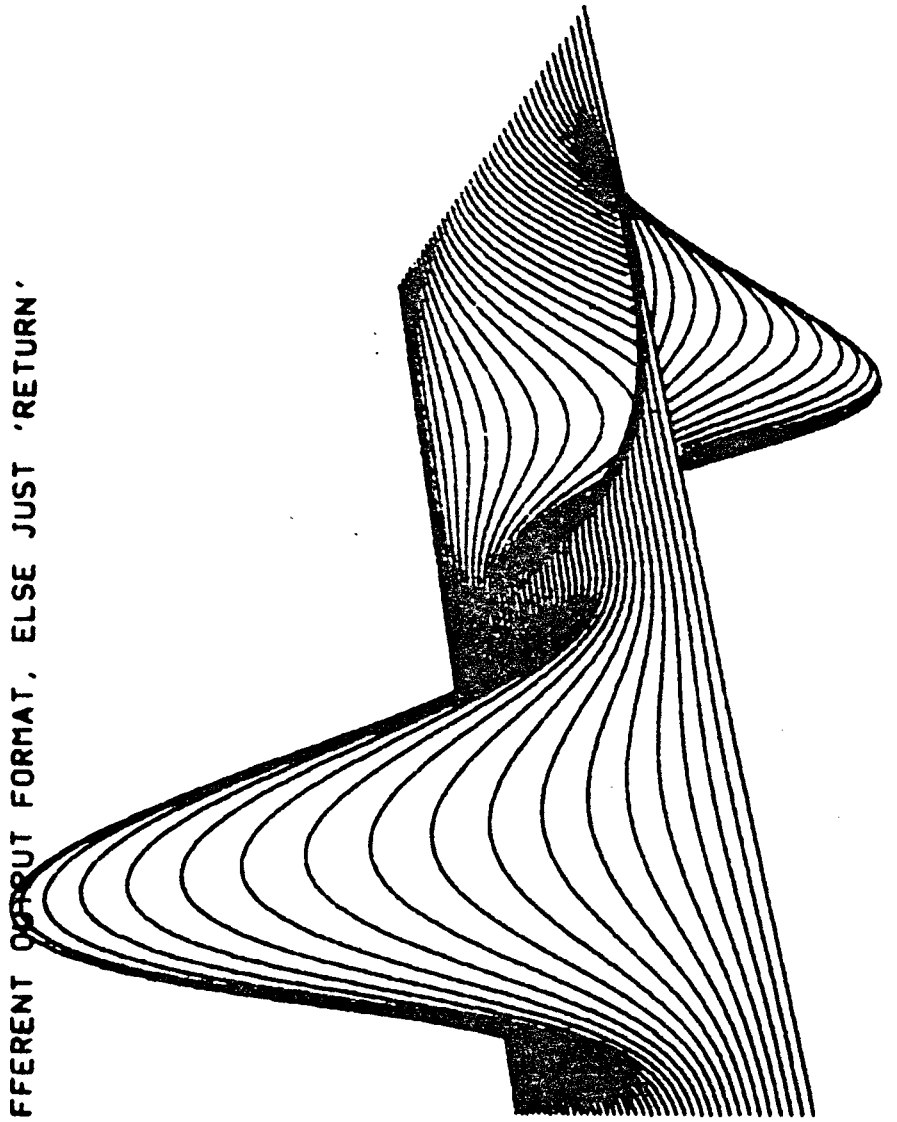
When complete you'll be asked whether to repeat the same drawing with different parameters (e.g., plotter instead of screen), whether a perspective from a closer or more distant viewpoint is required, or from a greater or smaller altitude of view. Surfaces can be drawn from underneath.

For a red/green stereo pair, Pen 1 on the 4663 should have a red pen, Pen 2 a green pen. The success of the 3-D image seen when wearing red/green spectacles depends greatly on the form of the surface and on the viewpoint chosen.

```
PROGRAM STEREO SURFACE PLOTS A FUNCTION
CONTAINED IN LINES 3010 TO 3990.

THE PRESENT FUNCTION IS

3000 REM SUBROUTINE TO EVALUATE FUNCTION
3010 Z=3*SIN(X)↑3*SIN(Y)↑3+0.5
4000 RETURN


TO CHANGE THIS, STOP THE PROGRAM AND REWRITE THESE LINES,
GIVING THE VARIABLE Z A VALUE DEPENDENT ON VARIABLES X & Y.

PRESS 'RETURN' TO CONTINUE


NOW ENTER X AND Y LIMITS:
X MINIMUM: -3
X MAXIMUM: 3
Y MINIMUM: -3
Y MAXIMUM: 0

ENTER 1 FOR SINGLE DRAWING ON PLOTTER
      2 FOR STEREO PLOT
      OR JUST 'RETURN' FOR SCREEN OUTPUT
```

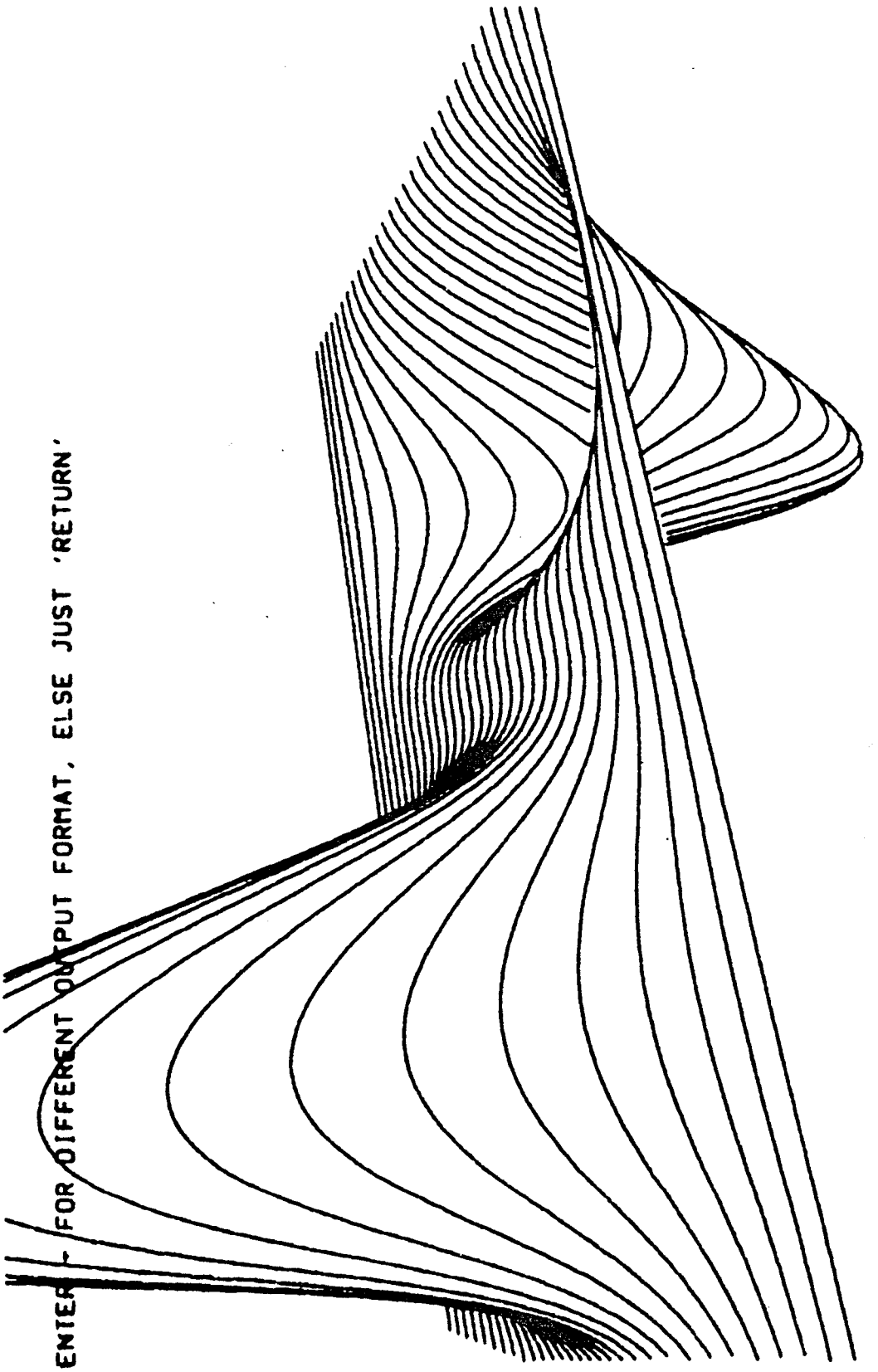
TITLE

Stereo Surface



TITLE

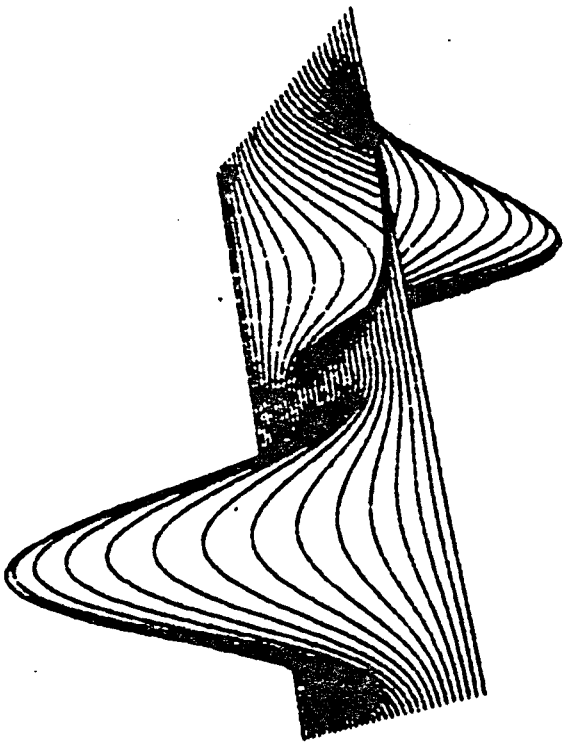
Stereo Surface



TITLE

Stereo Surface

ENTER + FOR DIFFERENT OUTPUT FORMAT, ELSE JUST 'RETURN'



Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		ABSTRACT NUMBER
3-D Plot w/wo Hidden Lines		
ORIGINAL DATE	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
March, 1981		16K
AUTHOR		PERIPHERALS
Lothar Tschimpke	Munich, Germany	Optional - 4662/4663 Plotter

ABSTRACT

Files: 1 ASCII Program

Statements: 272

This program plots a net or lines of a mathematical or programmable function of two variables, e.g., Z=F(X,Y). The function can be viewed under any angle.

When using the Hidden-Line Algorithm, the Z-axis has to be vertical.

The program automatically begins drawing in front and generates a pattern, which gives the information, whether next line can be drawn or not.

TITLE

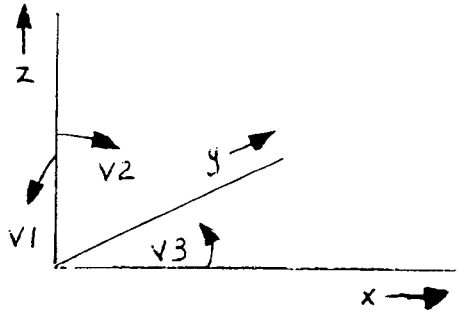
3-D Plot w/wo Hidden Lines

OPERATING INSTRUCTIONS

The user provides the following information in the mainline program:

N	Number of points of one line.
NØ	N/NØ lines are to be drawn. NØ = 1: each line; NØ = 2: each second line; etc.
QØ	Selection of lines in X- or Y-direction; QØ = 0: lines parallel to the X-axis; QØ = 1: lines parallel to the Y-axis.
Q2	Q2 = 1: without Q2 = 2: with hidden line algorithm.
Q3	Q3 = 0: test whether actual line is above or below the last lines; Q3 = 1: test only whether actual line is above the last lines.
Q4	Q4 = 1: for equal dimensions of X- and Y-limits; Q4 = 2: for different dimensions of X and Y (e.g., ft. and miles)
V1	Slope angle of the X-Y plane.
V2	Slope angle of the Z-axis (has to be Ø for hidden-line alg.).
V3	Angle of the X-Y plane.
Z1	Display address for the plotting device in use. for the 4050 screen: Z1 = 32
P	The X,Y,Z-coordinates are used in this program as P(1),P(2),P(3).
Q(2,3)	Minimum and maximum limits of the range to be drawn.

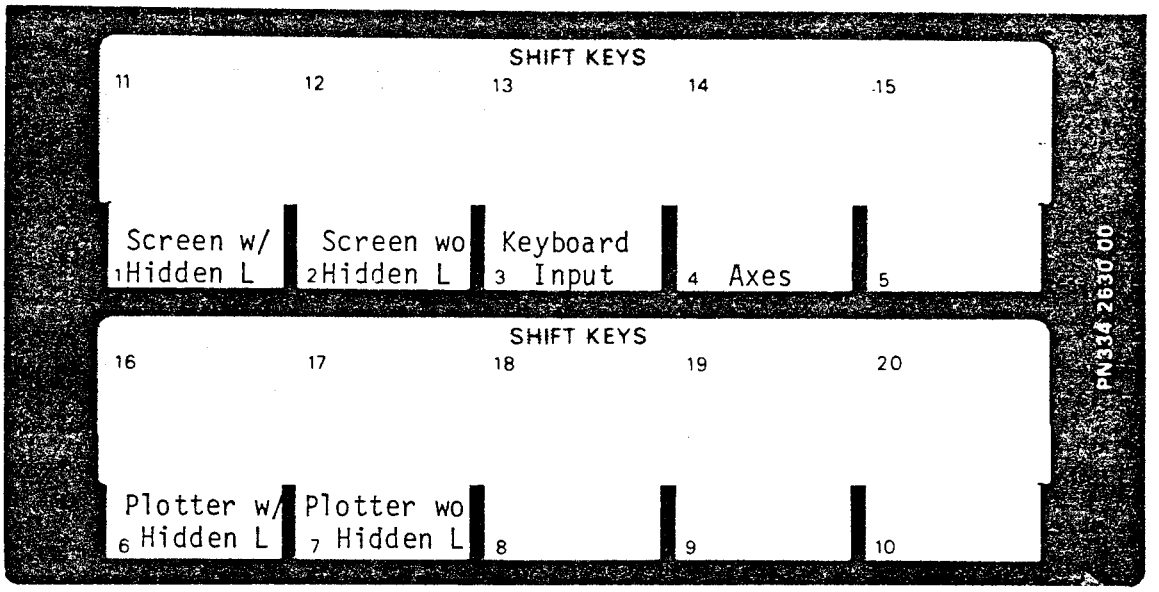
Angles of view:



TITLE

3-D Plot w/wo Hidden Lines

TITLE TAPE # FILE #



LOADING INSTRUCTIONS

Load the program into memory through the tape directory, or FIND 35 and OLD.

Change the statements in lines 5020-5070 to reflect your chosen function.

Key in RUN.

TITLE

3-D Plot w/wo Hidden Lines

SHORT DESCRIPTION AND USE FOR ANY FUNCTION

The program calculates the X- and Y- coordinates, named P(1) and P(2), in the area which is given by the limits of Q(2,3) and presents it to the subroutine in line 5000. There the value of Z, named P(3), is calculated as a function of P(1) and P(2). Any mathematical or programmable function may be positioned in the lines following 5000 closed by a RETURN statement.

These coordinates are automatically transformed to the planar coordinates XØ and YØ which can be executed by a DRAW or MOVE statement.

Bibliography:

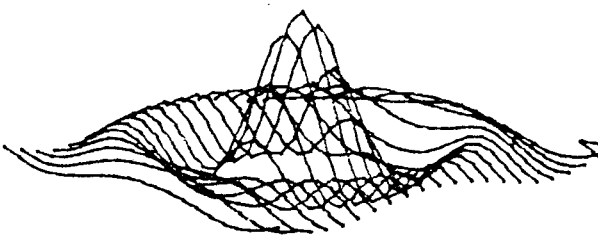
3-D with Perspective
Will Gallant
Applications Library 51/00-9507/0

USED VARIABLES

I	I1	I2	J	N	NØ	P	Q	QØ	Q2
Q3	Q4	S	SØ	T	TØ	V	VØ	V1	V2
V3	V4	V5	V6	V7	V8	V9	X	XØ	X1
X2	X3	X4	X5	X6	X7	X8	X9	YØ	Y1
Y2	Z	Z1							

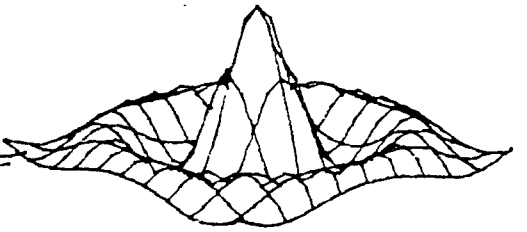
TITLE

3-D Plot w/wo Hidden Lines

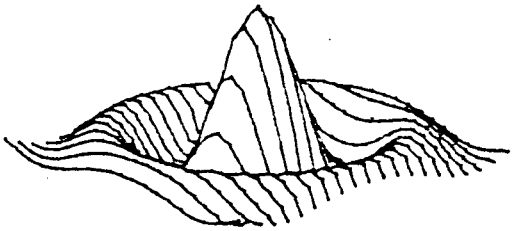


N=2Ø

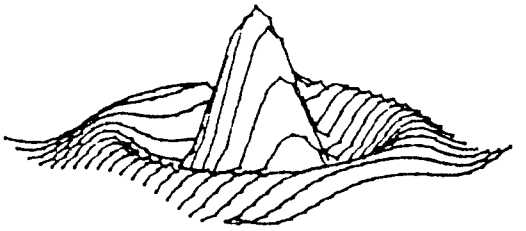
QØ=Ø
Q2=1
Q3=Ø
NØ=1



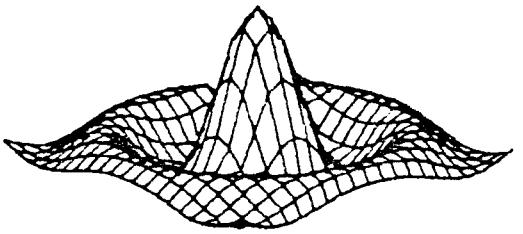
QØ=Ø,1
Q3=Ø
NØ=2



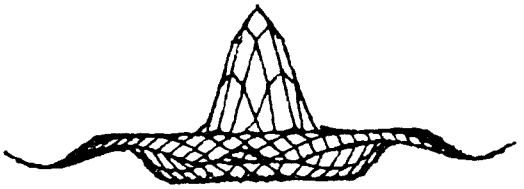
QØ=Ø
Q2=2
Q3=Ø
Q4=1or2



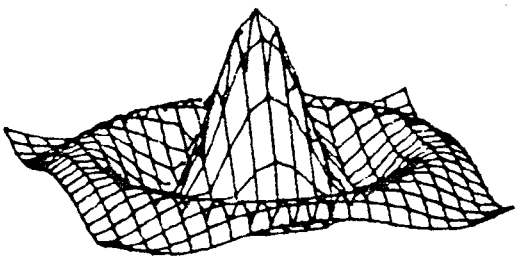
QØ=1
Q2=2
Q3=Ø
Q4=1or2



QØ=Ø,1
Q2=2
Q3=Ø
Q4=1or2



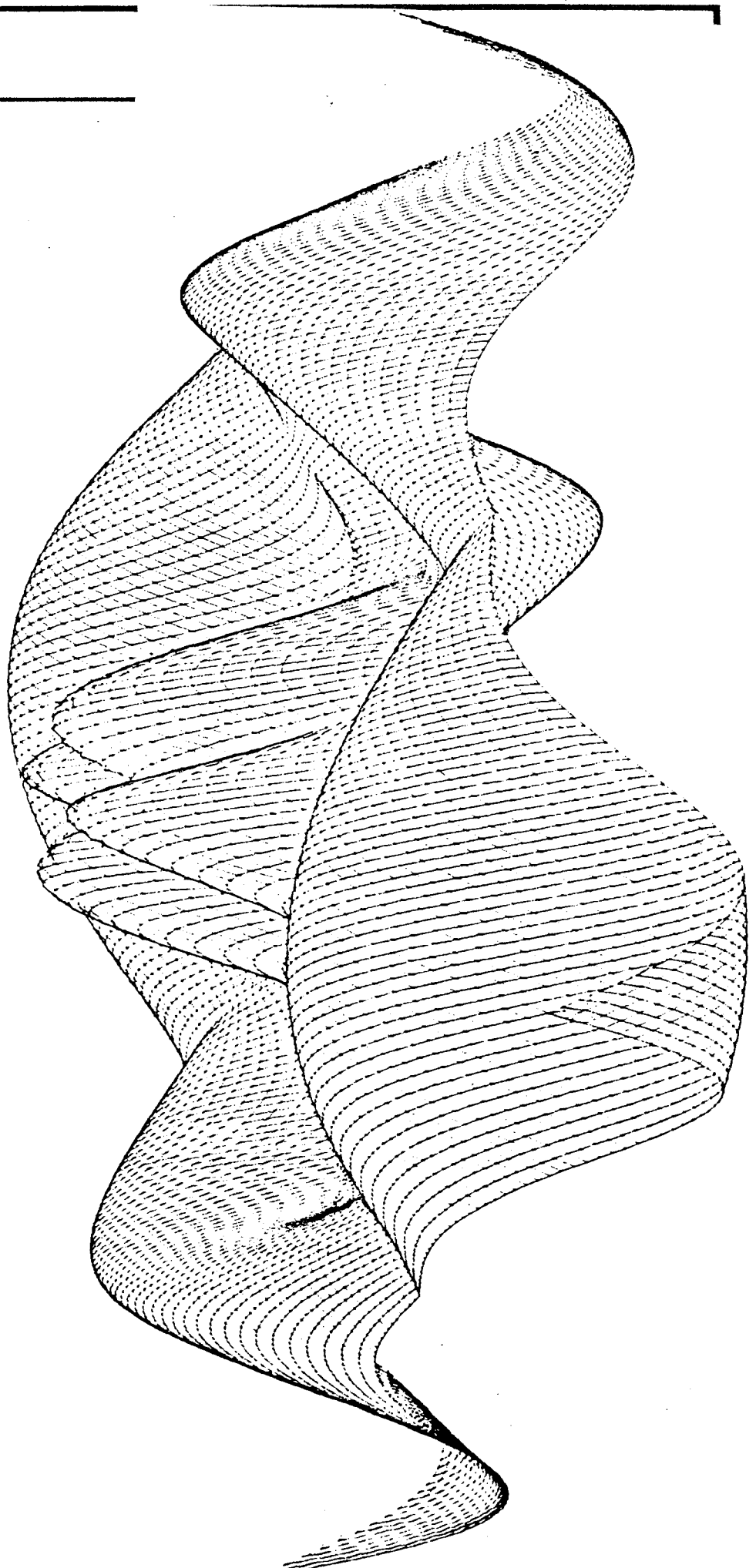
V1,V2,V3=Ø,Ø,-5Ø



17,Ø,-7Ø

TITLE

3-D Plot w/wo Hidden Lines



L. Tschimpke
Werner-Heisenberg-Weg 39
D-8014 NEUBIBERG
GERMANY

OOC

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	
PLOT3D	
ORIGINAL DATE / REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
	See Below 1
AUTHOR Robert Kennedy c/o Elliot Noma	PERIPHERALS
3501 Market St., Phila,, Pa. 19104 (215)386-0100	See Below 2

ABSTRACT

PLOT3D is a plotting routine which is to be run as an appended subroutine to a main program. PLOT3D receives a vertical z values for a sequence of points on a line through the x-y plane. These coordinates are rotated on the x and z axes and plotted in the two dimensional plane on the graphic display unit with a hidden line routine.

---

1 4051
16k (option 20)

2 4907

4662 (can be used with program modification)

TITLE

PLOT3D

Description

PLOT3D receives z values for points on a line on the X-Y plane. The x,y,z coordinates for these points are rotated around the z-axis and then around the x-axis. PLOT3D then plots the line segments joining the rotated points, provided the segment appears above the hidden line horizen stored as a mask. The mask is an array which spans the screen's horizontal axis containing the values of the maximum vertical values (in screen coordinates) for the particular horizontal value.

The plotted point is checked against the mask by comparing vertical values for the point and the mask given by the horizontal values. (NOTE: for this reason the horizontal values plotted must be positive, in the range of the mask, and integers.) The nine possibilities of this comparison are shown in the chart below:

Current Point	Previous Point	Result
> mask	> mask (=mask)	PLot line between points
> mask	< mask	Find intersection of mask and the line and draw from there
= mask	> mask (= mask)	Plot line between points
= mask	< mask	Nothing
< mask	> mask	Find intersection of line and mask, draw to there
< mask	< mask (= mask)	Nothing

Once plotted or attempted to plot, the mask values for the horizontal values affected is updated.

After the line is plotted, control returns to the calling program and PLOT3D waits for the next line of z values.

Internal Data Storage

Variable	Used to Store	Type
F2	Flag to bypass initializing calculations	Simple
Z, N	Number of line	Simple
K0	Number of point	Simple

TITLE

PLOT 3D

Internal Data Storage contd.

Variable	Used to Store	Type
K1	Loop parameter	Array (2)
K3	Loop step	Simple
X1	X value for current point	Simple
Y2	Y-value for current point	Simple
A1 thru A5	Rotation factors	Simple
Q	Sent Y-value	Array (N8)
SØ	Shift factor for Y	Simple
F8	Flag showing the status of previous point	Simple
X2	X-value for previous point	Simple
Y2	Y-value for previous point	Simple
C1	X-value for intersection point	Simple
C2	Y-value for intersection point	Simple
C3, C4	Used to calculate intersection point	Simple
M	Mask	Array (2000)
M1	Slope of plot line	Simple
M2	Added value to adjusted mask	Simple
MO	Mask value for current point	Simple
P1	Rotation on Z-axis	Simple
T1	Rotation on Y-axis	Simple
S1	Shift value for X	Simple
N8	Number of points per line	Simple
F1	Flag showing which plot program is on (first or cross-hatch)	Simple

TITLE

PLOT3D

Internal Data Storage contd.

Variable	Used to Store	Type
lo	Estimated minimum y-value	simple
Hl	Estimated maximum y-value	Simple

Methods

See Description Page 1

Update of Mask- The slope between the previous point and current point is calculated and is added to the mask value of the horizontal points between the other two points in question

Intersection of line and mask- The intersection of these two lines is achieved by a formula derived from the standard point-slope formula of plane geometry

Rotation of Axes- Five rotation factors are derived from the sine and cosine of the given angles and are applied to the appropiate cooridinate

Cross-Hatching- After the first plot, the x-axis is rotated an additional 90 degrees and the coordinates are changed in retrospect by the main program; the vertical values are sichronized at an arbitray point and the horizontal values at zero

Operating Instructions

1. Old main program which appends PLOT3D*

2. Insure all inputed data is received

3. RUN

Eg. OLD "sinxy", "a"
RUN
(the program is now interactive)

*On the TEKniques Vol. 5 No. 4 T1 tape, load through the program directory, or FIND 36 OLD and RUN.

NOTE: If the program files are transferred to occupy other locations on a tape, statements 190 and 210 in file 36must be changed.

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
Pipechart		
ORIGINAL DATE March 1981	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED 16K
AUTHOR Steven Salisbury Whirlpool Corporation Benton Harbor, MI		PERIPHERALS 4662 Plotter - optional

ABSTRACT

1. Description

This program quickly provides a three dimensional bar chart with shading. Up to 12 bars may be displayed on the 4050 screen or the 4662 plotter,

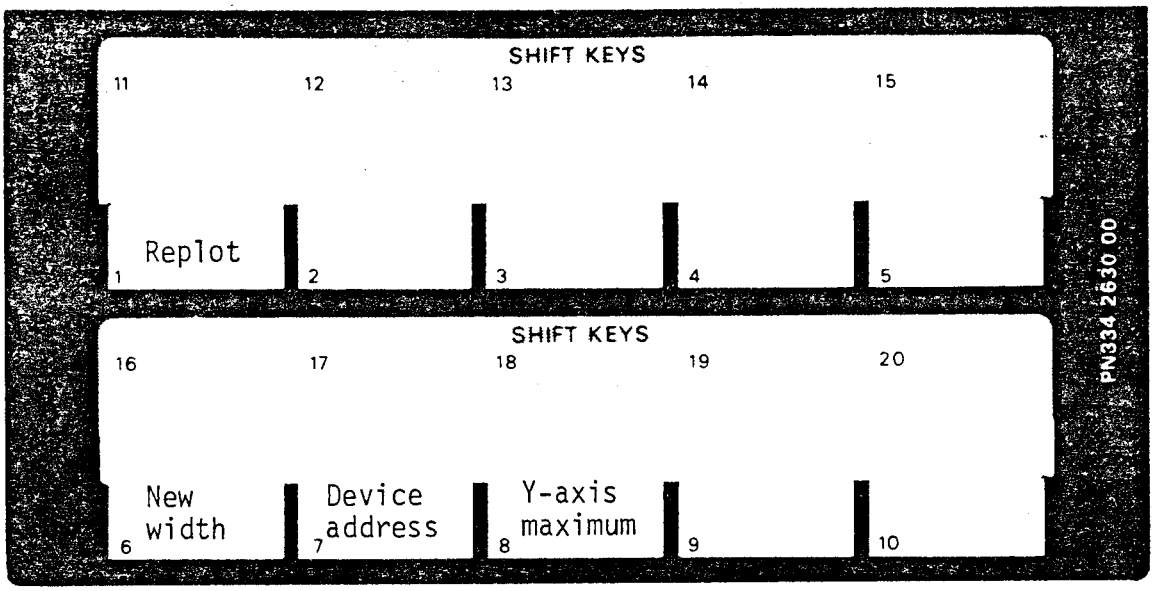
Data entered from the keyboard is displayed in the form of N cylindrical bars, where N is the number of bars selected by the user. The bars are drawn on a stand, and a title block is placed above the bars for a one line, 52 character title. Due to the intensity of the shading, the program does not shade bars when drawing on the plotter.

The Y-axis minimum is always set to zero. Only positive values may be plotted

TITLE

Pipechart

TITLE Pipechart TAPE # FILE #



4. Operating Instructions

a. Overlay

Key #	Function
1.	Enters program at beginning of plot section.
6.	User selects new width, and width is corrected for number of bars.
7.	User selects new address for plot of chart
8.	User selects new Y-axis maximum, Tic mark interval is updated also (max./5).

Note: Keys are used after 1st. execution to allow user to alter aspects of the graph without re-entering information.

b. Program loading

Since this is a stand alone program, it may be placed on any tape or disc file. From TEKniques Vol. 5 No. 4 T1 tape, load through the directory, or FIND 39, and OLD.

c. Program execution

Upon startup of the program, the following information is asked for:

1. Number of bars (1-11)
2. Bar width (0-1)
3. Output device address (1 or 32)
4. Title (52 character maximum)
5. Data values for each bar

TITLE

Pipechart

2. Internal Data Storage

Variable	Used to Store	Type
N	Number of Bars	Simple
Q	Output Device Address	Simple
W	Bar Width	Simple
A$	Title	String A$(52)
A	Data	Array A (N)
M	Y-axis maximum	Simple
P	Y-tic interval	Simple
O	Y-tic minimum (always zero)	Simple
W1	Flag to draw full or half ellipse	Simple
X	Horizontal ellipse position locator	Simple
Y	Vertical " " "	Simple
X1	Horizontal ellipse range	Simple
Y1	Vertical " "	Simple
X2	Horizontal screen position for each bar	ARRAY X2 (N,2)
Y2	Vertical " " " " "	ARRAY Y2 (N)
X4	Current horizontal shading position	Simple
Y4	" vertical " "	Simple
C	Vertical shading counter	Simple
X5	Next horizontal shading position	Simple
Y5	" vertical " "	Simple
D1,D2	Dummy input variables	Simple
I,J,J1,K	For/next loopers	Simple

3. Method

This program uses both UDU &GDU capabilities to develop the ellipses necessary to give the three diminsional appearance of the graph. It locates the positon of the ellipses by use of UDU's, then viewports a given area (determined by the number and width of bars) and draws the ellipse. As the full ellipse is drawn, the bar is shaded. This is done by a series of GDU input, move and draw statements.

The stand is arbitrarily placed at the bottom of the bars, and the title block is placed above the bar area. Tic marks are printed along the side of the bar area and corresponding scallops (half ellipses) are drawn on the bars.

TITLE

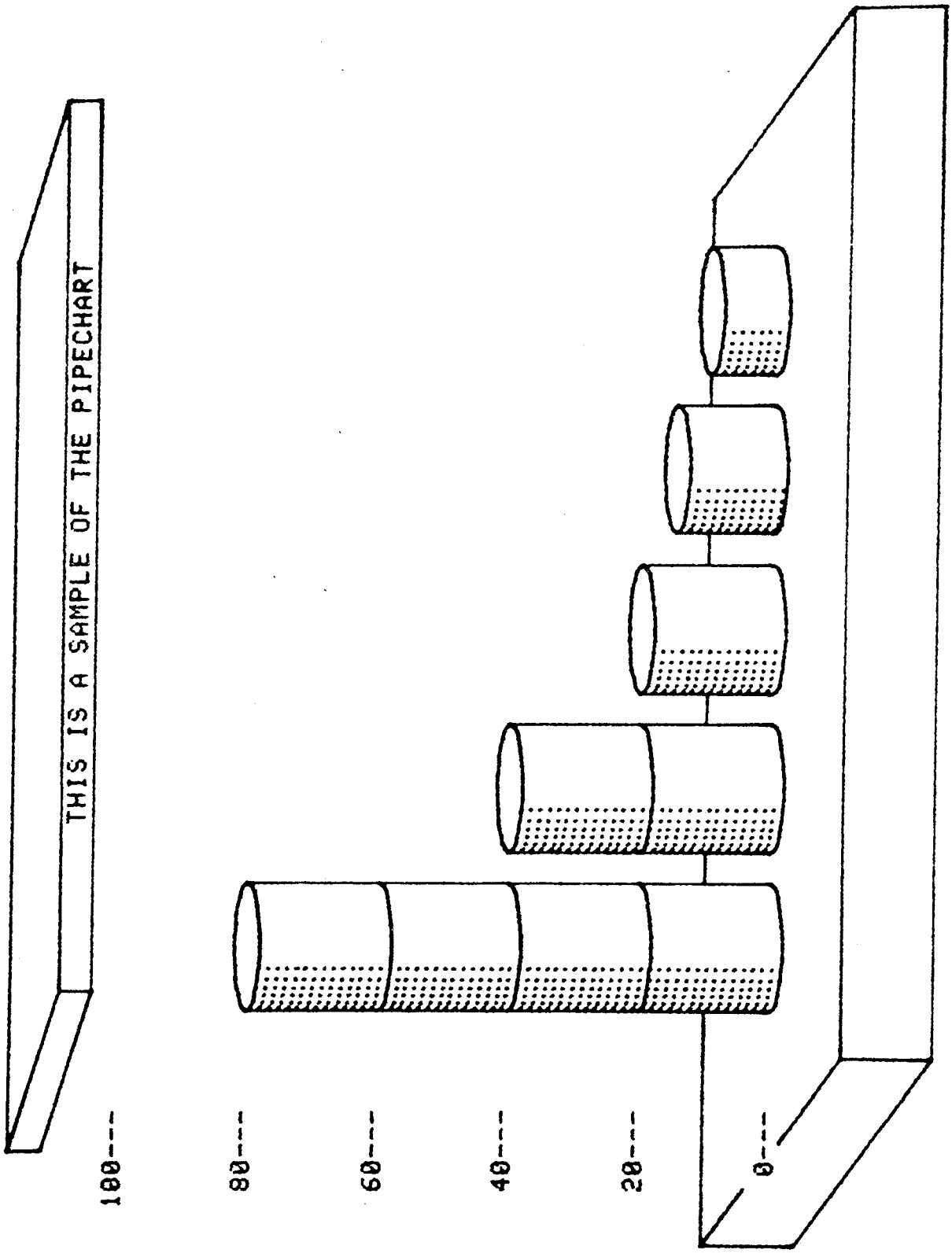
Pipechart

The graph is drawn, and the system goes out of execution mode. At this time, the user may select any user definable keys to alter aspects of the graph and re-draw it.

Two examples follow
Listing enclosed.

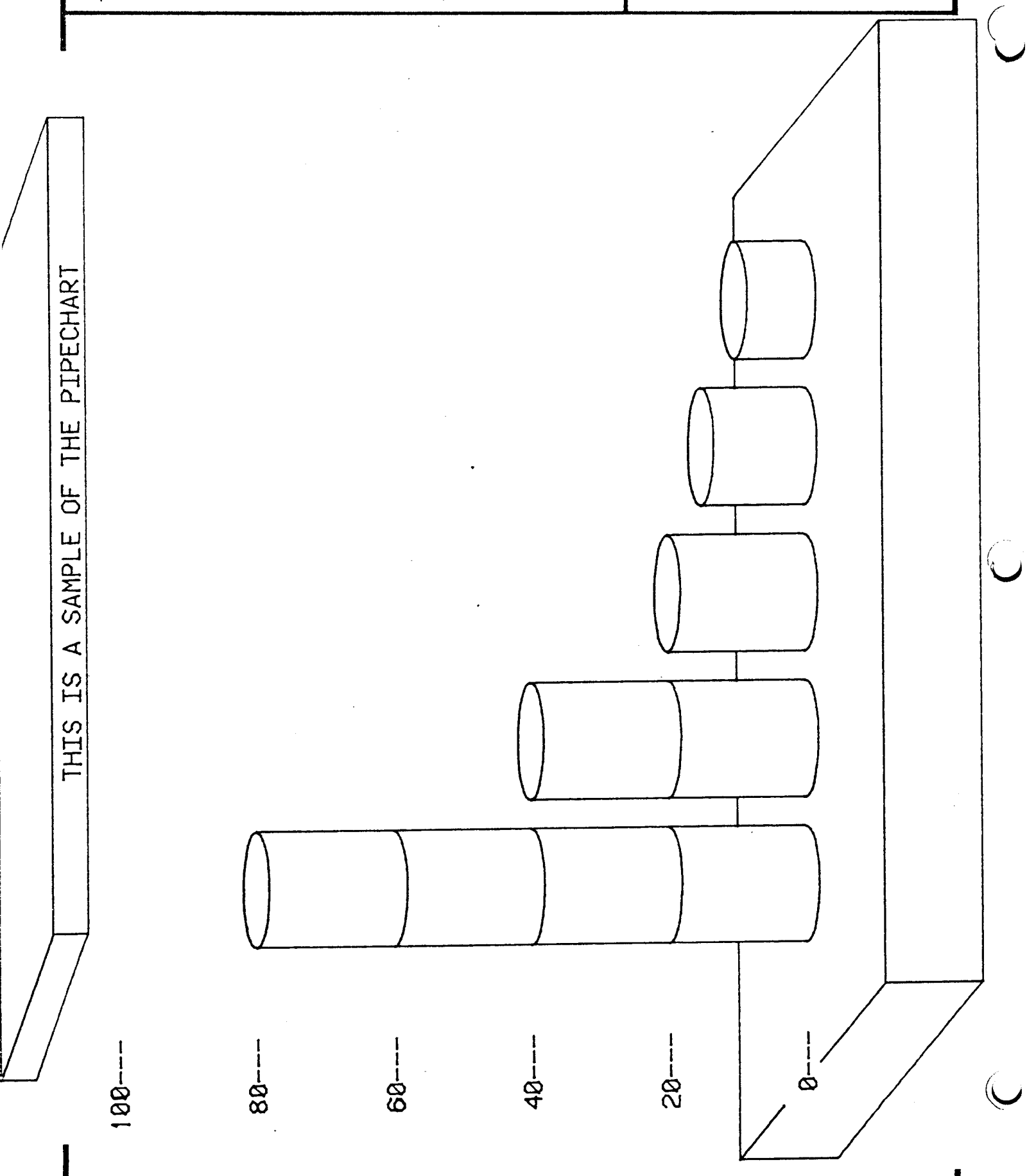
TITLE

Pipechart



TITLE

Pipechart



Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
Contour Plots		
ORIGINAL DATE	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
1/21/81		8K
AUTHOR		PERIPHERALS
Jerry W. Anderson 12 D1 Phillips Building Bartlesville, OK 74004 (918) 661-7438		Optional 4662, 4631

ABSTRACT

This program plots response surfaces (or contour plots) in two dimensions. It uses the 3-dimensional full-quadratic model (or subsets thereof) as a function:

Z = YØ + A*X + B*Y + (AB)*X*Y + (ASQ)*X ↑ 2 + (BSQ)*Y ↑ 2

where Z, X, Y are variables and YØ, A, B, AB, ASQ, & BSQ are constants which may be known or determined by regression. Very high resolution plots may be generated. The user may determine the number of contours desired and their value.

In effect, we have displayed a three-dimensional data space in two dimensions, much as a topographical map displays geographical features.

TITLE

Contour Plots

For a fixed contour value Z, and for increments over the range of the x axis, we may solve for Y using the quadratic formula. We then connect consecutive (x, y) pairs. Consideration is also given to the case in which the discriminant (b^2 - 4 ac) <0.

5. Operating instructions

a) Overlay - none is used
b) Program Loading
From TEKniques Vol. 5 No. 4 T1 tape, load through the tape directory, or FIND 40, OLD and RUN.

c) Program Execution - Answer these prompted questions:

1) Enter output device #
2) Enter degree of resolution desired
3) Enter plot titles
4) Enter # contours desired
5) Enter contour values
6) Enter the constants from the full quadratic model (be sure to enter a zero when appropriate)
7) Enter Axis information (Minimum, maximum, increment)

Example:

Consider the equation:

$$Z = 1 + X + Y + XY + X^2 + Y^2$$

over the range $X \in (-20, 20)$
$Y \in (-20, 20)$

Look at contour values of Z equal to (10, 20, 30, 40, 50, 60, 70, 80, 90, 100).
Using high resolution, we may see the results on the following page.

The effect you see is much the same as looking from above into an oval dish.

TITLE

Contour Plots

2. Data Tape Structure

No tape or disc files are required by the program.

3. Internal Data Storage

Variable	Strings
N1 = Width Parameter in Pri @ T, 17:	C$ = String of "*"
N2 = Height Parameter in Pri @ T,17:	A$ = Plot Title
T = Output Device #	H$ = Horizontal Axis Title
P4 = Contours	V$ = Vertical Axis Title
R (P4) = Vector of contour values	X$ = Scratch String
YØ = Constant Term	
A = x Coefficient	
B = y Coefficient	
C = xy Coefficient	
A2 = x ↑ 2 Coefficient	
B2 = y ↑ 2 Coefficient	
M = Degree of Resolution	
N7 = Plot min, x axis	
N8 = Plot max, x axis	
N9 = Plot increment, x axis	
M2 = Plot min, y axis	
M3 = Plot max, y axis	
IØ = Plot increment, y axis	
I = For/next Loop Index	
J = For/next Loop Index	
K = For/next Loop Index	
N = For/next Loop Index	
KØ = Loop Counter	
P1 = Starting index of plot loop	
Q = Stopping index of plot loop	
S = Loop Step (±1)	
E = Drawing increment (UDU)	
X = x axis plotting values (current)	
Y = y axis plotting values (current)	
P = b^2 - 4 ac from quadratic equation	
O1 = x axis plotting values (previous)	
O2 = y axis plotting values (previous)	
W = Width of plotter page	
L = Length of plotter page	

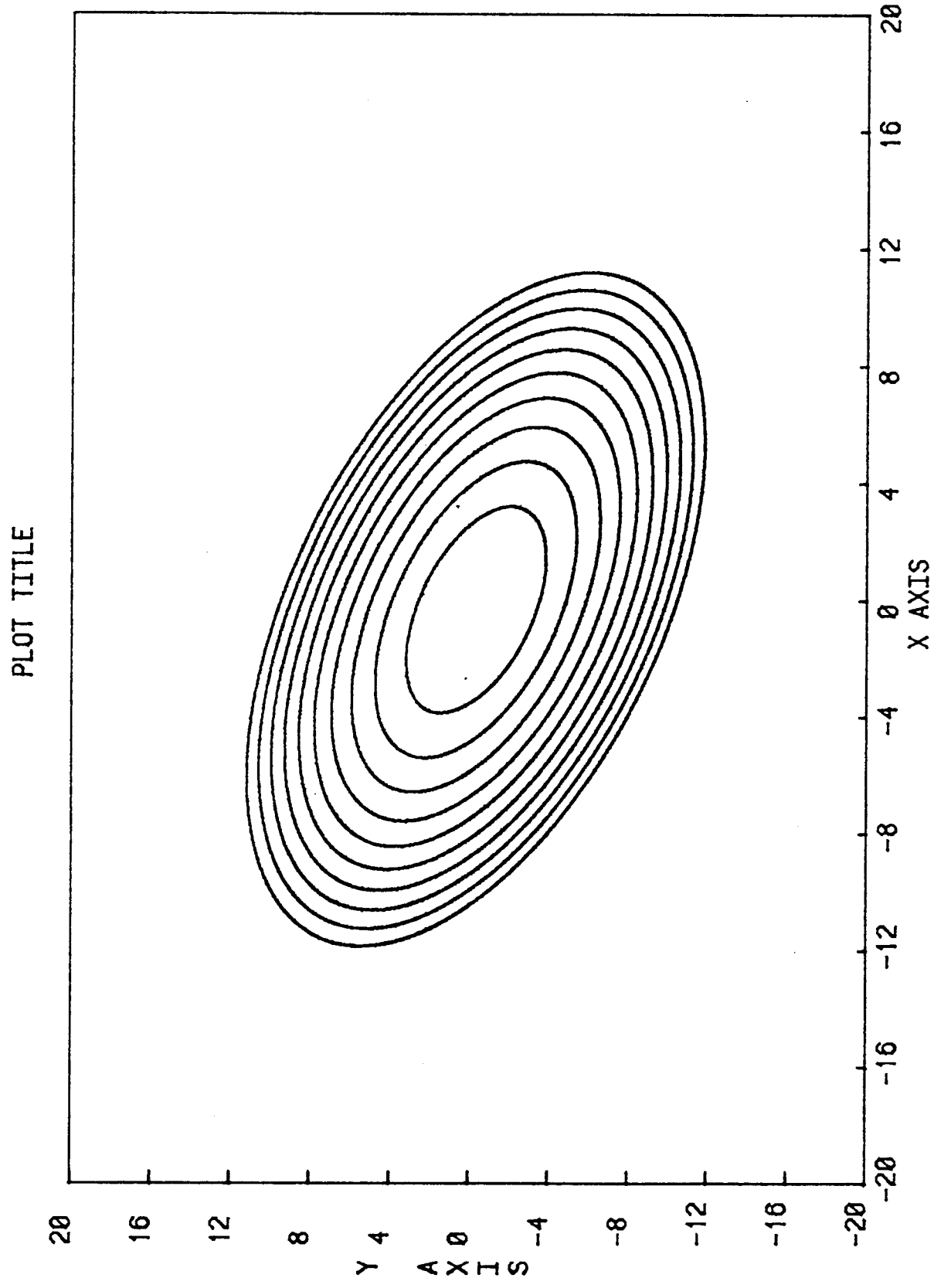
4. Methods

It uses the quadratic formula

$$y = \frac{-b \pm \sqrt{b^2 - 4\,ac}}{2a}$$ to solve $ay^2+by+c = 0$.

TITLE

Contour Plots



Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	
Hierarchal Clustering	
ORIGINAL DATE: 1/20/81 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 32K
AUTHOR Jerry W. Anderson 12 D1 Phillips Building Bartlesville, OK 74004 (918) 661-7438	PERIPHERALS: OPT 4662, 4631

ABSTRACT

Consider a data set consisting of N vectors (or data points) each in L dimensions.

A sequence of classifications in which larger clusters (or groupings) are obtained through a merger of smaller ones is called a nested or hierarchal classification. Basically, it shows the relationship between each data point and every other point in the data set by forming a tree or dendrograph (similar to a family tree). The similarity of two data points is inversely related to the distance between them along the branches of the tree.

There are basically three types of hierarchal clustering: single linkage, average linkage, and complete linkage. Among these, single linkage is the most simple. It begins by finding the link between the two closest data points. The succeeding stages consist of finding the shortest remaining link which directly joins together data points not already joined indirectly through a chain of links or branches. In essence, it searches through the distance matrix seeking the shortest distance that will contribute to further linkage or branching of the clusters.

TITLE

Hierarchal Clustering

2. Data Tape Structure

   A. Data may be either ASCII or Binary. It is stored rowwise as a data matrix.

   B. The file to be used must be specified by the user.

   C. It must have been previously created prior to running this program (using Editor or a Data Entry program).

   D. Data may exist on program tape or a separate data tape.

   E. Keyboard Entry is an option.

3. Internal Data Storage

X	Data Matrix	Array (N,M)
D	Distance Vector	Array (I9)
I1	Index Vector	Array (I9)
I2	Working Vector of Cluster #'s	Array (N)
I3	Ordered Cluster Vector	Array (N)
I4	Merge Vector	Array (2N-2)
D1	Merge Distances	Array (N-1)
X1	Tree Horizontal Coordinates	Array (N)
Y1	Tree Vertical Coordinates	Array (N)
N	Rows in Data Matrix	Scalar
M	Cols in Data Matrix, Counter	Scalar
I9	N*(N-1)/2	Scalar
V	Output Device #	Scalar
N1	N-1	Scalar
N2	2*N	Scalar
N4	N2-2	Scalar
F	File # of Data	Scalar
Y$	Answer String	String
Q9	TYP of File F	Scalar
C	Mean Vector	Array (M)
V1	Standard Deviation Vector	Array (M)
S	Counter	Scalar
I	For/Next Index	Scalar
K	For/Next Index	Scalar
B	Scratch	Scalar
K1	Scratch	Scalar
P1	Scratch	Scalar
T1	Scratch	Scalar
J	Scratch	Scalar
M1	Scratch	Scalar
M2	Scratch	Scalar
M3	Scratch	Scalar
M4	Scratch	Scalar
K	Scratch Counter	Scalar
L2	Scratch Index	Scalar
L1	Scratch Index	Scalar

TITLE

Hierarchal Clustering

4. Methods: Single Linkage Hierarchal Clustering Algorithm -

I. Compute the Lower Triangular Distance Matrix.

II. Store the results as the triplets (d_{ij}, i, j), where d_{ij} is the distance between data point i and data point j according to some suitable distance metric or measure.

III. Sort this collection of triplets into ascending order on d_{ij}.

IV. The first merger corresponds to the first distance in the sorted list, i.e., the smallest distance in the whole distance matrix.

V. Choose for the next merger that pair of data points with the next largest distance, provided the two data points are not already in the same cluster (i.e., connected through one or more links). If the data points are already in the same cluster, bypass the merger and move on to the next triple.

VI. Continue Step V until m-1 mergers have been made, where m is the number of data points in the data set.

5. Operating Instructions

A. No Overlay is used.

B. To load the program, from TEKniques Vol. 5 No. 4 T1 tape, load through the tape directory, or FIND 41, OLD and RUN.

C. 1. The Program is executed by answering the necessary questions as prompted from the screen:

a) # rows, # columns in data set
b) Row identifiers (2 characters/Row)
c) Device Address of Output
d) File # for data (Ø = Keyboard Entry)
e) Desire to center & Scale Data?

The answer is always "yes" unless your measurements (columns) are all measured on the same scale.

f) For 4662 output, enter paper size.

2. Examples (data is on file 44)

Eleven measurements were made on each of 8 products. Products B and Y were actually different lots of the same product, as were I & J, L & M, and C & E. We desire to know something about the relative similarities and differences among these eight products. The data matrix is as follows:

B:	2022170	25607	21199	46.5	4.1	4.0	0.0034	0.0025	460	160	10
Y:	2070000	22300	16356	47.0	4.0	4.0	0.0016	0.0015	520	159	42
C:	2414685	13007	9729	66.6	5.3	4.6	0.0470	0.0320	450	270	182
E:	2261855	16254	11731	66.6	5.3	4.5	0.0600	0.0330	430	270	182
I:	2251560	16037	11774	55.5	5.5	5.4	0.0064	0.0043	310	240	15
J:	2227055	16617	12310	50.4	5.5	5.4	0.0070	0.0045	280	240	7
L:	2442670	19619	14601	54.4	5.1	5.0	0.0079	0.0049	500	250	111
M:	2456880	16530	13195	53.3	5.2	5.1	0.0063	0.0036	470	250	133

TITLE

Hierarchal Clustering

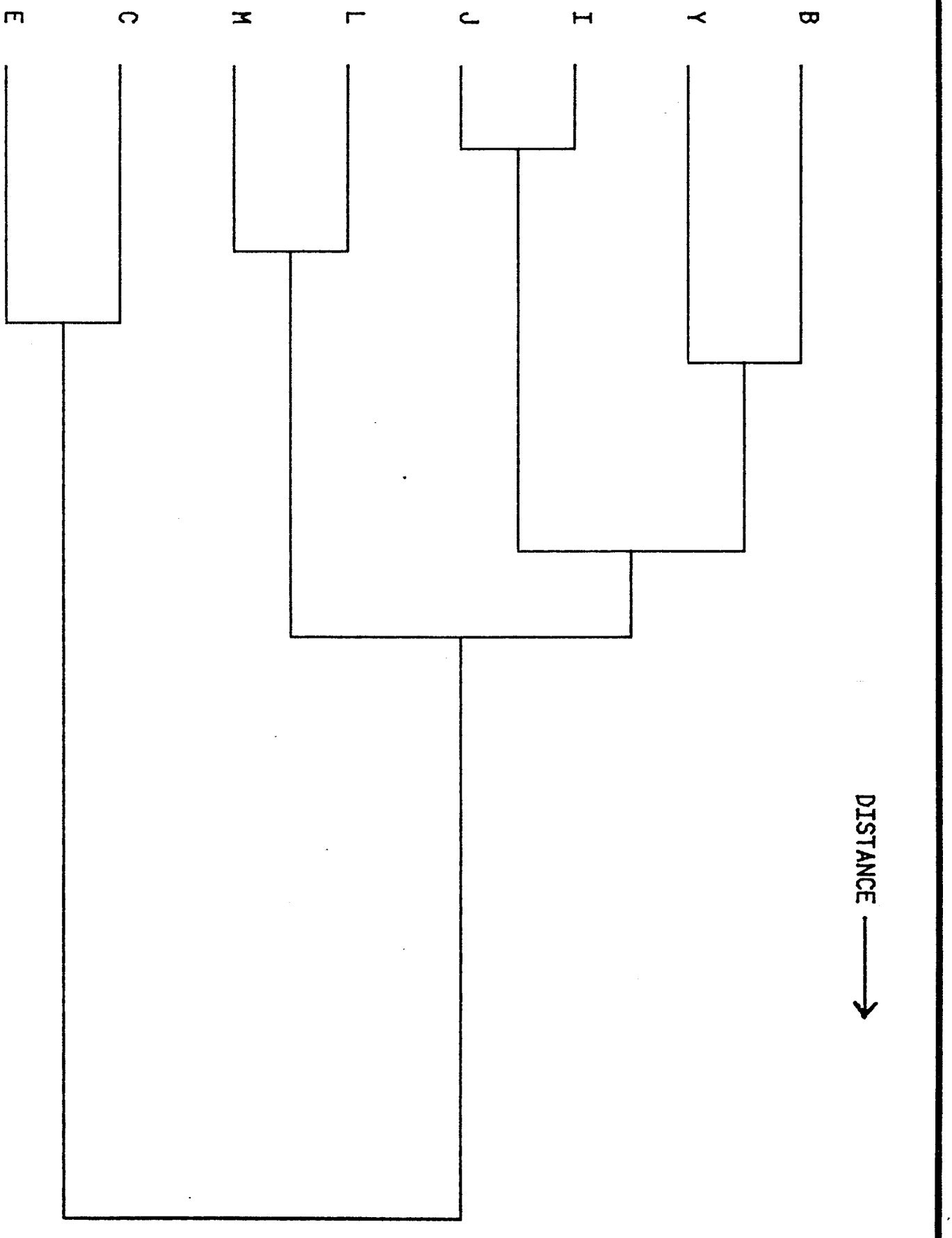
The output of the program may be seen on the following page. We may conclude that I & J are the most similar products made, followed by L & M, C & E, then B & Y. The grouping into pairs was as expected. (The smaller the distance with which two products "join up", the greater the similarity.) We may then conclude that among all pairs, pair B & Y and pair I & J are most similar. We then see that the group B, Y, I, J, L, & M are more similar among themselves than among the pair of C & E.

6. References

Anderberg, Michael R., Applications of Cluster Analysis New York Academic Press, 1973.

TITLE

Hierarchal Clustering



TITLE

Hierarchal Clustering

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	
Nonlinear Mapping	
ORIGINAL DATE: 1/21/81 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 32K +
AUTHOR: Jerry W. Anderson, 12 D1 Phillips Building, Bartlesville, OK 74004 (918) 661-7438	PERIPHERALS: Optional 4662, 4631

ABSTRACT

1. Nonlinear Mapping

The purpose of nonlinear mapping is to reduce the dimension of a data set consisting of, say, N vectors with L dimensions to one of lower dimensions. Specifically, it is usually desired to reduce the dimensionality to 2 so that a cross plot may be used to observe the groups (or clustering) of the data units. This mapping to a lower-dimensional space is made such that the data structure is approximately preserved. This preservation is maintained by obtaining N points in lower-dimensional space such that their interpoint distances approximate the corresponding interpoint distances in L-space. In order words, we seek a set of N 2-dimensional vectors whose distance matrix (N x N) is very near the distance matrix (N x N) of the original data set, i.e., N vectors in L dimensions.

Designate the N vectors in L-space by X_i, i = 1, ...,N. Designate the N vectors in d-space (where d = 2 usually) by Y_i, i = 1, ...,N. Let the distance (according to any desired distance metric) between X_i and X_j in L-space be defined by $d_{ij}^* = \text{dist}(X_i, X_j)$. Similarly, let the distance between the corresponding vectors Y_i and Y_j in d-space be defined by $d_{ij} = \text{dist}(Y_i, Y_j)$.

We shall choose starting values for the Y values in d-space. Using these starting values, we shall calculate the d-space distance, d_{ij}, from which we define an error E. This error tells us how well the present configuration of N points in d-space fits the N points in L-space:

$$E = \frac{1}{\sum\{d_{ij}^*\}_{i<j}} \sum^{N} \frac{\{d_{ij}^* - d_{ij}\}^2}{d_{ij}^*}$$

The next step in this iterative algorithm is to adjust the Y variables so as to decrease the error E. A steepest descent procedure is used in order to minimize the error.

TITLE

Nonlinear Mapping

2. Data Tape Structure

Data may be ASCII or Binary. The data matrix must be stored rowwise on a single previously-created file. The data may be entered with the Editor Rom Pack or with any Data Entry Program. Data may be on program tape or a separate data tape.

3. Internal Data Storage

Arrays

D (M, N) = Data or observation matrix
D1 (M, M) = Upper triangular distance matrix
T (M,U) = Solution matrix
C (N) = Centroid vector
V (N) = Variance vector
C1 (M,U) = First partials
C2 (M,2) = Second partials

Strings

Y$ (1) = Answer string to standardize data
A$ = Plot title
H$ = Horizontal axis title
V$ = Vertical axis title
I$ = String of data point product code
S$ = Product codes string

Scalars

N1 = # iterations
A = Fudge factor empirically determined (.3 or .4)
U = # of dimensions we reduce to
M = # Rows in data matrix
N = # Columns in data matrix
M1 = M-1
F = File # of data
T5 = Device for graphical output
W9 = Width of plotter page
L9 = Length of plotter page
W4 = Type of File F
S = Scratch sum
P = For/Next index
Q = For/Next index
I = For/Next index
I1 = I + 1
J = For/Next index
K = For/Next index
B = Max. Column Variance
I8 = Column with max. variance
I9 = Column with next largest variance
Z = 1.0E-5 Fuzz
L = For/Next index
D2 = Scratch partial
A1 = Scratch
F1 = Scratch
F2 = Scratch
F3 = Scratch
T9 = Distance Update
S = Counter
E = Mapping error
XØ = x Min. for plot
X1 = x Max. for plot
YØ = x Min. for plot
Y1 = Y Max. for plot
N9 = x Axis increment
IØ = Y Axis increment
Y9 = Length of I$
N7 = Number iterations

TITLE

Nonlinear Mapping

4. Method:

Nonlinear Mapping Algorithm

Let E(m) be the mapping error after the m^{th} iteration:

$$E(m) = \frac{1}{c} \sum_{i<j}^{N} \{d_{ij}^* - d_{ij}(m)\}^2 / d_{ij}^*$$

$$\text{where } c = \sum_{i<j}^{N} \{d_{ij}^*\}$$

$$\text{and } d_{ij}(m) = \sqrt{\sum_{k=1}^{d} \{y_{ik}(m) - y_{ik}(m)\}^2}$$

Algorithm: The new d-space configuration at time m+1 is given by

$$y_{pg}(m+1) = y_{pg}(m) - MF * \Delta pg(m)$$

$$\text{where } \Delta pg(m) = \frac{\partial E(m)}{\partial y_{pg}(m)} \Big/ \left| \frac{\partial^2 E(m)}{\partial y_{pg}(m)^2} \right|$$

and MF ≃ .3 or .4 (this was determined empirically)
The partial derivatives are given by:

$$\frac{\partial E}{\partial y_{pg}} = \frac{-2}{c} \sum_{\substack{j=1 \\ j \neq p}}^{N} \frac{\{d_{pj}^* - d_{pj}\}}{d_{pj} d_{pj}^*} (y_{pg} - y_{jg})$$

$$\text{and } \frac{\partial^2 E}{\partial y_{pg}^2} = \frac{-2}{c} \sum_{\substack{j=1 \\ j \neq p}}^{N} \frac{1}{d_{pj}^* d_{pj}} \{(d_{pj}^* - d_{pj}) - \frac{(y_{pg} - y_{jg})^2}{d_{pj}} (1 + \frac{d_{pj}^* - d_{pj})}{d_{pj}}\}$$

TITLE

Nonlinear Mapping

In essence, we designate the N vectors in L-space by X_i, i = 1, ...,N. Designate the N vectors in d-space (where d = 2 usually) by Y_i, i = 1,...,N. Let the distance (according to any desired distance metric) between X_i and X_j in L-space be defined by $d_{ij}^* = \text{dist}(X_i, X_j)$. Similarly, let the distance between the corresponding vectors Y_i and Y_j in d-space be defined by $d_{ij} = \text{dist}(Y_i, Y_j)$.

We shall choose starting values for the Y values in d-space. Using these starting values, we shall calculate the d-space distance, d_{ij}, from which we define an error E. This error tells us how well the present configuration of N points in d-space fits the N points in L-space.

$$E = \frac{1}{\Sigma\{d_{ij}^*\}i} \sum_{<j}^{N} \frac{\{d_{ij}^* - d_{ij}\}^2}{d_{ij}^*}$$

The next step in this iterative algorithm is to adjust the Y variables so as to decrease the error E. A steepest descent procedure is used in order to minimize the error.

5. Operating Instructions

a) No overlay is used
b) Find x, where x is the file #
On TEKniques Vol. 5 No. 4 T1 tape, load through the tape directory or FIND 42, OLD and RUN

c) Answer the appropriate prompted questions:

1) Size of data matrix
2) # iterations desired (40 usually adequate)
3) Device address for output
4) Enter size of plotter paper on 4662, if used.
5) Enter the dimension to which you want data reduction (must be 2 to observe data graphically).
6) Enter file # for data set. Data may be either ASCII or Binary and stored rowwise.
7) Answer if you desire data centered and scaled (i.e., normalized). In most cases, the answer will be yes. (The only exception would be if all columns in your data set have the same "units" and same scale.) As in most multivariate techniques, failure to normalize will result in "big" numbers carrying more weighting than smaller ones.

TITLE

Nonlinear Mapping

Example: (data is on file 44)

Consider the following data matrix consisting of 11 measurements on each of 8 products labeled B, Y, C, E, I, J, L and M:

B:	2022170	25607	21199	46.5	4.1	4.0	0.0034	0.0025	460	160	10
Y:	2070000	22300	16356	47.0	4.0	4.0	0.0016	0.0015	520	159	42
C:	2414685	13007	9729	66.6	5.3	4.6	0.0470	0.0320	450	270	182
E:	2261855	16254	11731	66.6	5.3	4.5	0.0600	0.0330	430	270	182
I:	2251560	16037	11774	55.5	5.5	5.4	0.0064	0.0043	310	240	15
J:	2227055	16617	12310	50.4	5.5	5.4	0.0070	0.0045	280	240	7
L:	2442670	19619	14601	54.4	5.1	5.0	0.0079	0.0049	500	250	111
M:	2456880	16530	13195	53.3	5.2	5.1	0.0063	0.0036	470	250	133

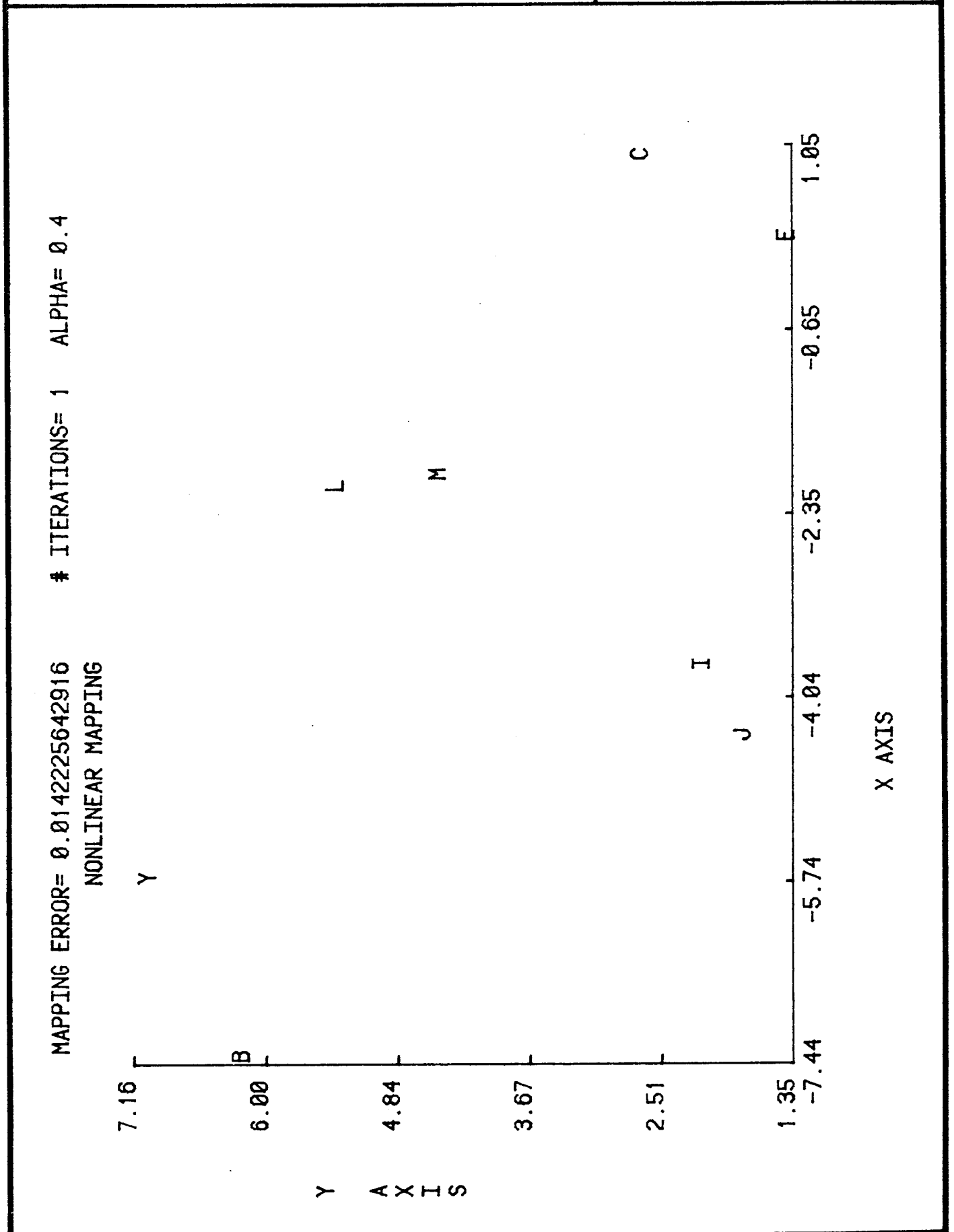
We wish to know the relative similarities and differences among the products with respect to all measurements as a group. By using nonlinear mapping as a data reduction technique, we can reduce this data set from (8, 11) to (8, 2) (while preserving interpoint distances as much as possible) and then crossplot the two resulting columns. Closeness on this crossplot indicates relative similarity. For example, as may be seen on the following page, products B & Y are very similar, as are C & E, I & J, and L & M. However, the pair B & Y is relatively dissimilar to the pair C & E. The mapping error of .014 is good. Further discussion may be seen in the reference.

6. References

Sammon, John W., Jr. "A Nonlinear Mapping for Data Structure Analysis." IEEE Trans. Comput. Vol. C-18 pp. 401-409, May 1969.

TITLE

Nonlinear Mapping



Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		EQUIPMENT AND OPTIONS REQUIRED
Vertical Plot		
ORIGINAL DATE: 1/21/81	REVISION DATE:	8K
AUTHOR: Jerry W. Anderson 12 D1 Phillips Building Bartlesville, OK 74004 (918) 661-7438		PERIPHERALS: Optional 4662, 4631

ABSTRACT

This program graphically presents on one page a data set consisting of M rows in N dimensions. Each of the M observations is represented by a 1-character code, which is plotted on each of N vertical scales across the page. A typical use of the program would be in an examination of M products, each with N measurements or properties (missing values are allowed). Relative product differences, with respect to each of the N measurements, are easily observed on the scales. Product and measurement labeling is allowed.

Thus, we have depicted in two dimensions an N dimensional data space. We have consolidated many plots on one page.

TITLE

Vertical Plot

2. Data Tape Structure

A. Data may be either ASCII or Binary in one data file. This file must have been previously created (and the data matrix stored rowwise) through any data entry program or through the "Editor". Data may be on the same data tape or on a separate tape.

B. Product Codes (optional) must be stored in an ASCII data file. Enter one product code per record.

3. Internal Data Storage

Arrays

B(M,N)	Array holding data matrix
M3 (N)	Vector of column minimums
M4 (N)	Vector of column maximums
F9 (N)	Column heading lengths

Simple

T	Device address of output
F	File # of data matrix
M	# rows in data matrix
N	# columns in data matrix
M9	Missing value indicator #
K1	Character width
M1	Scratch column minimum
M2	Scratch column maximum
A7	x minimum window
A8	x maximum window
K2	Tolerance for offsetting a character
K	For/Next Index
I	For/Next Index
J	For/Next Index

Strings

S$	- String of column headings
A$	- 1 character product labels
Y$	- Plot heading
R$	- Scratch string
Q$	- Product code scratch string
G$	- Answer string
B$	- Individual product label
Z$	- Scratch string holding column minimum, maximum

TITLE

Vertical Plot

4. Method

We operate on one column of the data matrix at a time. On a vertical scale, we plot each data point appropriately on the scale from minimum to maximum. We label it with the appropriate 1-letter code, being careful not to overwrite a letter that has been previously printed.

5. Operating instructions

A. No overlay is used
B. Find x, where x is the correct file
On TEKniques Vol.5 No. 4 T1 tape, Load through the tape directory or FIND 43, OLD and RUN.

Answer appropriate questions.

C. Answer the prompted questions:

1) Device # for output
2) Enter product labels
3) File # of data matrix
4) Size of data matrix

Missing value number - any real number may be used to represent a missing value, which will be ignored.

5) Enter heading for Plot
6) Enter product code file # (optional)
7) You may choose to set your own minimums and maximums for each scale.

Example: (Data is on File 44)

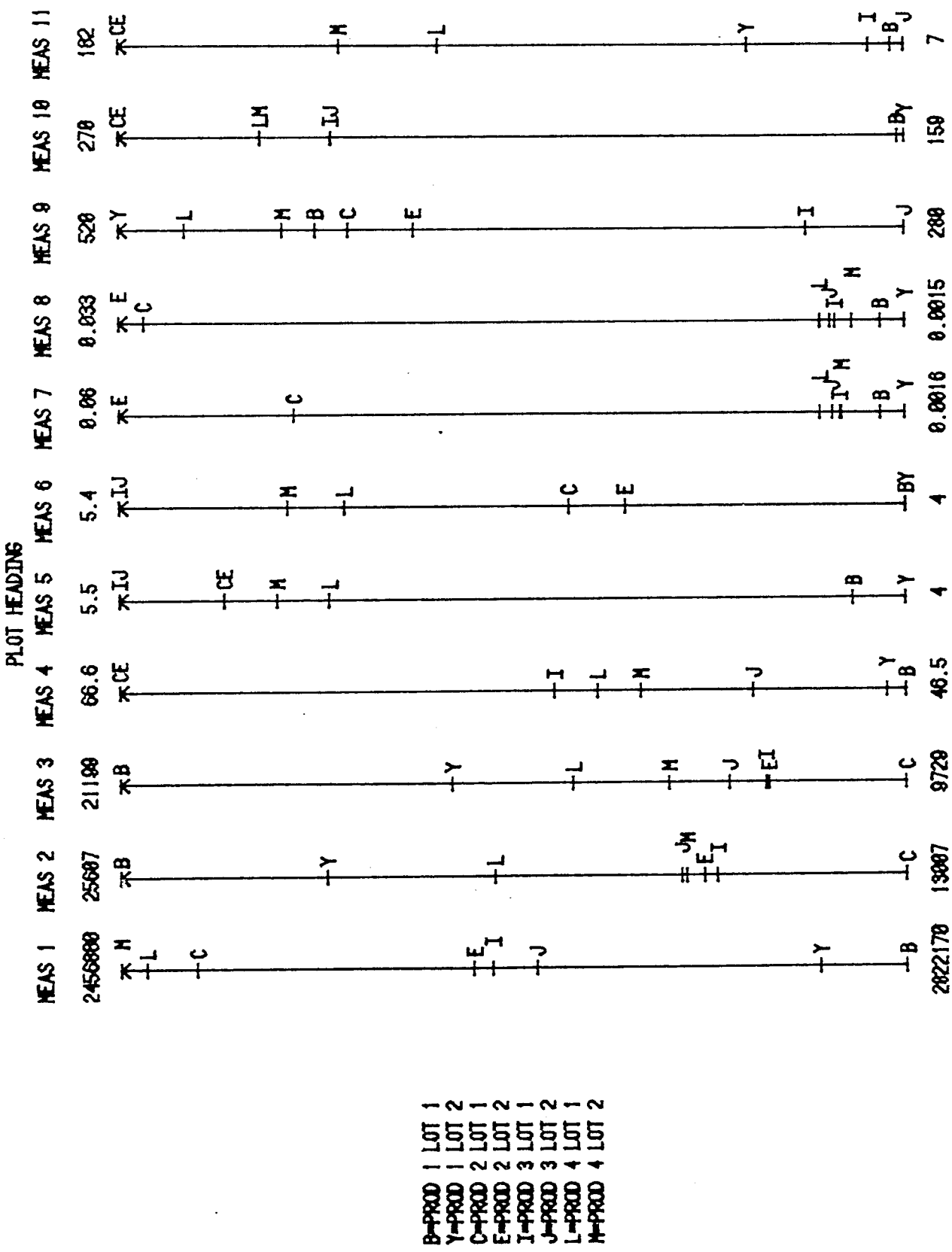
Consider the 11 measurements made on each of 8 products (labeled B, Y, C, E, I, J, L, & M) as listed below:

B:	2022170	25607	21199	46.5	4.1	4.0	0.0034	0.0025	460	160	10
Y:	2070000	22300	16356	47.0	4.0	4.0	0.0016	0.0015	520	159	42
C:	2414685	13007	9729	66.6	5.3	4.6	0.0470	0.0320	450	270	182
E:	2261855	16254	11731	66.6	5.3	4.5	0.0600	0.0330	430	270	182
I:	2251560	16037	11774	55.5	5.5	5.4	0.0064	0.0043	310	240	15
J:	2227055	16617	12310	50.4	5.5	5.4	0.0070	0.0045	280	240	7
L:	2442670	19619	14601	54.4	5.1	5.0	0.0079	0.0049	500	250	111
M:	2456880	16530	13195	53.3	5.2	5.1	0.0063	0.0036	470	250	133

We may observe the program output on the following page. We see, for example, that with respect to measurement 10, products C & E had high values (270) whereas products B&Y had low values (≈160). We may make this comparison for each of the 11 measurements. Product comparisons & selections are thus made easy.

TITLE

Vertical Plot



Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		
REFERENCE INDEX		
ORIGINAL DATE	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
March, 1979		32K
AUTHOR		PERIPHERALS
F.G.G. Voigts	National Physical Res Lab Council for Sci & Ind Res Pretoria, South Africa	

ABSTRACT

Files: 1 ASCII Program
Requires Dedicated Tape

Statements: 385

This program creates, stores on tape, updates and lists an index of literature references. The following data may be stored:

Author
Journal
Volume
Date
Page
Status (or type) Symbol
Title

The minimum information required is author, year of publication and the status symbol. The rest is optional and the fields could be used for other information.

The program assigns a code to each reference, comrpised of the first three letters of the author's surname, the year of publication and another letter to distinguish between otherwise equal combinations. The references are arranged alphabetically in 26 data files according to these codes. New references are sorted and inserted in the appropriate file. Existing references within a file may be deleted or changed.

TITLE

Reference Index

CONTENTS

TITLE

Reference Index

1. DESCRIPTION

References are entered in the following way:

Author/Journal/Volume(Date)Page/Status(Type)Symbol
Title

Only the author, year of publication and the status (type) symbol are mandatory, the rest is optional and the fields could be used for other information.

The program assigns a six character code to each reference, as follows:

AAAnnX where:

AAA - first three letters of the author's surname

nn - year of publication

X - letter to distinguish between otherwise equal codes.

The program arranges the references alphabetically according to the codes assigned to each.

The main program consists of a number of subprograms which are called by the User-Definable Keys (UDK). These programs will be described briefly. The functions of the UDKs will become clear from the operating instructions in Chapter 4. Flow diagrams are given in appendix A. The variables used by the program are listed in Appendix B.

Initialize (UDK 1)

This subroutine initializes the program, and must be executed before any of the other subprograms can be run. This routine is called by pressing UDK 1 (Initialize) or UDK 6 (Input).

The main functions of this subroutine are:

- delete string and array variables;
- assign dimensions to string and array variables;
- set initial values;
- set environment;
- determine whether titles are to be entered (printed).

TITLE

Reference Index

Input Control (UDK 3/UDK 6)

This program controls the reference input cycle. It is run after (or instead of) "Initialize" by pressing UDK 6 (Input), or can be started without deleting data by pressing UDK 3 (Continue Input).

The following steps are contained in a loop which ends only when 150 references have been entered or when a UDK is pressed:

- call input subroutine
- display the entered reference
- add the reference to L$.

Input Subroutine (UDK 3 through 6)

This subroutine enters references and compiles a temporary code. The program assigns an A as the last character in the code; it will be changed, if necessary, by the "Save Input" routine when the data is stored on magnetic tape (see Save Input). The routine is called by other subprograms whenever new reference data must be entered.

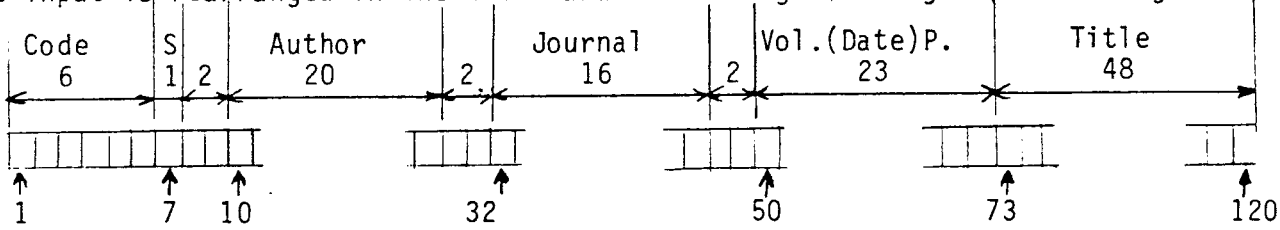
The user splits the entered references into fields using "/" (slash) as end-of-field delimiter. Any blank spaces following a slash are removed.

The program compiles the code from the first three characters in the author field and the two characters before the right bracket in the third field (year of publication). Any spaces directly before the bracket are removed. An A is placed in the 6th position of the code. For convenience, the status symbol is appended to the code.

The input is checked for the following:

- sufficient number of fields (at least three slashes);
- minimum length of author (at least three characters);
- if the author starts with an alphabetic character;
- if a status symbol (other than a blank space) is given;
- if any date (right bracket in third field) is entered;
- if more than one right bracket is entered in the third field;
- minimum length of date (at least two non-blank characters).

The input is rearranged in the 120 character string H$ using the following format:



where S is the status symbol.

TITLE

Reference Index

List Input (UDK 8)

This routine lists the new references that have been entered (in the original order).

Change Input (UDK 4)

This routine changes the entered references which are not yet stored on magnetic tape. The user identifies the reference to be changed by entering its temporary code.

The reference is displayed, if found. It can be deleted or be replaced, in which case the "Input Subroutine" is called. If a temporary code is duplicated, the next entry with the same code can be found by entering N (next).

Continue Input (UDK 3)

This routine continues "Input Control" after an interruption. The data already entered will not be deleted, as it would if input were resumed by pressing UDK 6 (Input).

The number of entries and the last reference will be displayed before "Input Control" continues.

Save Input (UDK 9)

Add new references to the data on magnetic tape with this program. This is accomplished as follows:

- new references are arranged alphabetically;
- they are stored in file 2 on tape;
- the files that have to receive new data and the number of entires for each file are established;
- for each of these files:
  - the old data is read from tape;
  - new references are inserted, comparing the codes;
  - if necessary, the last character (A) of the codes of the new entries are changed to ensure that each reference has a unique code.

List Index (UDK 10)

The index stored on tape can be listed with the option to make a hard copy.

All references are read from the magnetic tape and are displayed with or without title (as determined with "Initialize").

The input data string is not destroyed by this program.

TITLE

Reference Index

Change Index (UDK 5)

This program changes reference data on the data tape. The reference is identified by its six-character code.

- the file number is established from the first character of the code;
- the complete file is read in, overwriting all input data in the memory;
- one of the following changes can be made:
  - change a status symbol;
  - replace (add) a title;
  - replace a reference using the "Input Subroutine" - the old code is retained, and if no new title has to be entered, the old title is kept, too;
  - delete an entry
- the changed data are stored again on tape before this routine restarts.

2. PROGRAM LIMITATIONS

Input field length (characters):

	maximum	minimum
Author	20	3
Journal	16	0
Volume (Date) Page	23	4
Status (or Type)	1	1
Title	48	0

Maximum number of references:

Entered at a time:	150
Stored on magnetic tape:	2000 approximately

TITLE

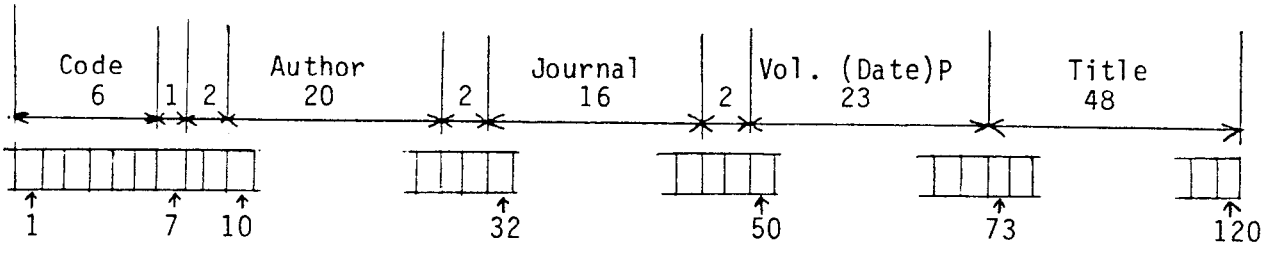
Reference Index

3. DATA TAPE STRUCTURE

The program is stored in file 1 which must be 9472 bytes long. File 2 (18432 bytes) is used as a scratch file. Files 3 through 28 contain the stored references arranged according to the first alphabetic character of the author's surname, starting with A in file 3. The lengths of the data files must be chosen according to the anticipated number of entries per character. The following division of the tape is recommended:

FILE #	LENGTH	CONTENTS	RECORDS
1	9472	Program	385
2	18432	Storage	150
3	10752	A	88
4	18432	B	150
5	14592	C	119
6	12288	D	100
7	6912	E	56
8	9984	F	81
9	13056	G	107
10	13824	H	113
11	3840	I	31
12	5376	J	44
13	13056	K	107
14	9216	L	75
15	16128	M	132
16	6144	N	50
17	6144	O	50
18	12288	P	100
19	4608	Q	37
20	11520	R	94
21	18432	S	150
22	7680	T	62
23	4608	U	37
24	7680	V	62
25	9216	W	75
26	3840	X	31
27	6144	Y	50
28	3840	Z	31

The references are stored in ASCII records of 120 characters per reference, using the following format:



TITLE

Reference Index

This allows about 31 references for surnames with the lowest frequency of occurrence (e.g., surnames with initial character X, Z, etc.) and a maximum of 150 references for those of the highest frequency of occurrence (e.g., B, S, etc.) to be stored on tape.

4. OPERATING INSTRUCTIONS

Introduction

Transfer the Reference Index program from the TEKniques program tape to the first file on a tape to be used solely for this program:

Insert TEKniques Vol. 5 No. 4 T1 tape

FIND 45

OLD

Insert your new tape

FIND 1

MARK 1,9472

FIND 1

SAVE

Loading and Running the Program

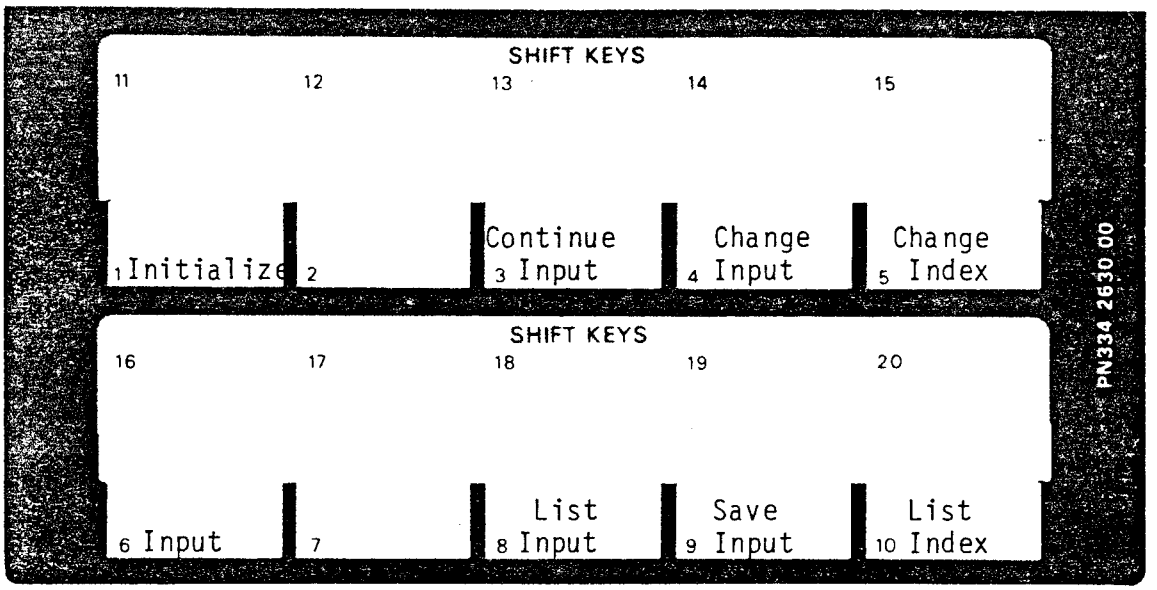
To load the program, insert your dedicated tape into the 4050 and press AUTO LOAD. Proceed as soon as the I/O light is off (note: the BUSY light will stay on since the machine is waiting for an instruction).

The different functions of the program are executed using the User-Definable Keys (UDK).

TITLE

Reference Index

TITLE TAPE # FILE #



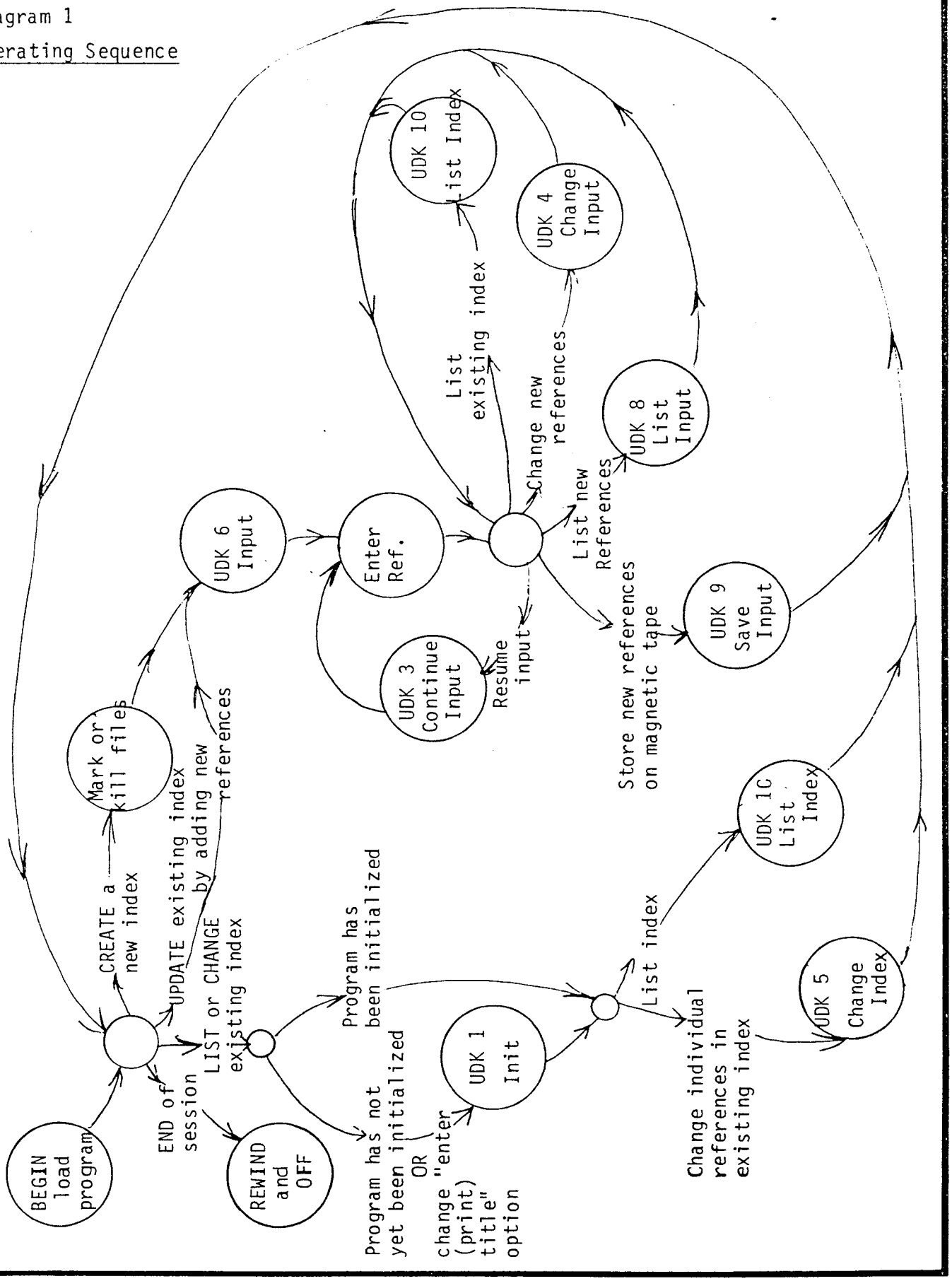
Use Diagram 1 (following page) to determine the sequence in which the UDK's may be used. Start in the upper left corner. Not adhering to the indicated sequence may result in loss of all data entered during the current session.

TITLE

Reference Index

Diagram 1

Operating Sequence



TITLE

Reference Index

USER DEFINABLE KEYS

UDK 1: Initialize

Press to initialize program when

- the index stored on tape is to be displayed or changed before any new references are entered;
- the "enter (print) title" option is to be changed.

A warning will be displayed:

```
INITIALIZE
WARNING:  THIS ROUTINE WILL DELETE ALL DATA
DO YOU WANT TO CONTINUE?  YES/NO  YES
```

Enter YES to proceed, thereby deleting all data in memory (not that stored on tape);

or

NO to exit from the routine without any loss of data in memory.

If YES was returned, the program responds:

```
DO YOU WANT TO ENTER (PRINT) TITLES?  YES/NO
```

Enter YES or NO. The answer will apply until changed by this or the "Input" routine (UDK 6).

UDK 2: Not used.

UDK 3: Continue Input

Press to resume input without loss of data after input interruption(s).

The number of references entered during the current session as well as the last entry will be displayed before the input loop starts.

TITLE

Reference Index

UDK 4: Change Input

Press to change data already entered during the current session.

Brief instructions will be displayed, followed by:

ENTER CODE OR "STOP"

Enter temporary code to select the desired reference for modification.

Enter STOP to exit from this routine.

If an existing code is entered, the first reference with that code will be displayed followed by:

ENTER INSTRUCTION (D, N or R)

Enter:

- D to delete the reference;
- R to replace the reference by another one, which must be entered according to the rules of "Input" (UDK 6).
- N to get the next reference with the same temporary code.

The routine restarts automatically after every change made, without again displaying the operating instructions.

UDK 5: Change Index

WARNING: This routine will delete all new references in memory.

Press to modify individual references stored on magnetic tape.

Brief instructions will be displayed followed by:

ENTER CODE OR "STOP"

Enter the six character code to select the desired reference for modification.

Enter STOP to exit from this routine.

TITLE

Reference Index

After an existing code has been entered, the reference will be displayed followed by:

ENTER INSTRUCTION: S[tatus], T[itle] , R[eference] or D[elete]

Enter

- S to change the status symbol;
- T to change the title;
- R to change the reference;
- D to delete the reference.

For a status or title change, only the new information will be requested.

A new reference must be entered according to the rules of "Input" (UDK 6). However, the old code from the reference being replaced will be retained together with the new reference. If the code is incorrect, refere to the correct one in the reference, e.g., in the "Journal" or "Title" field. Then enter correct code using "Input" (UDK 6) and "Save Input" (UDK 9). Or, simply delete the entry and later insert the new one (UDK 6).

The routine restarts automatically after every change made, without again displaying the instructions.

UDK 6: Input

Press to start input session.

The "Initialize" routine will be run first. For instructions, refer to UDK 1: Initialize.

Subsequently a summary of the input format will be displayed followed by a prompt:

FORMAT FOR REFERENCES:

AUTHOR/JOURNAL/VOL(DATE)PAGE/STATUS SYMBOL

- AUTHOR MUST START WITH A LETTER

- FIELDS ARE SEPERATED WITH /

- THE DATE MUST BE WRITTEN BETWEEN BRACKETS

- THE STATUS SYMBOL CAN BE ANY CHARACTER OR SYMBOL

ENTER REFERENCE

TITLE

Reference Index

Enter a reference using the correct syntax:

BECKER JA, ET AL./J CHEM PHYS/57(1974)157/.

The input must comply with the rules given below.

If the "enter (print) title" option was set to "yes", the next step applies, otherwise it will be skipped.

ENTER TITLE

Enter the title before the " | ". The reference will be displayed with the temporary code, followed by the next request for a reference.

Exit the input loop by pressing a UDK (see diagram 1).

Syntax rules for input:

Author/Journal/Volume(Date)Page/Status

The fields (author, journal, etc.) are separated by a "/" (slash). The first three slashes in the reference will be interpreted as end-of-field delimiters, and therefore no slashes may be used in the reference, except for the status symbol.

Author:

Write the first author's surname first. The first three letters in this field make up the beginning of the code. The author must start with an alphabetic character (A-Z). An author like F.T.S. Yu may be entered as Yu F.T.S.

Minimum length: 3 characters.

Maximum length: 20 characters.

Journal:

This information is optional.

Minimum length: 0 characters.

Maximum length: 16 characters.

TITLE

Reference Index

Volume (Date) Page:

The date must be written between brackets with the year of publication preceding the right bracket ")".

The minimum length of the date is 2 characters, not necessarily numbers, e.g., asterisks could be inserted for an unknown date. No other information in this field may be written between any brackets, i.e., (), [], {}.

All other information in this field is optional. The sequence Volume (Date) Page may be changed.

Minimum length: 4 characters.

Maximum length: 23 characters.

Status

Any character except a space must be supplied. It will be written immediately after the code in the listing.

Length: 1 character.

Title

There are no rules concerning the syntax.

Minimum length: 0 characters

Maximum length: 48 characters.

When the maximum length of a field is exceeded, that information will be truncated.

Some examples are given on the next two pages. (No titles were entered as the inclusion is trivial.) All entries on the first page are valid. The second page contains some incorrect entries. Note that the last reference was accepted although author and date were not entered according to the syntax rules. This resulted in an incorrect code.

UDK 7: Not used.

UDK 8: List Input

Press to display the entered references that are not yet stored on magnetic tape.

UDK 9: Save Input

Press to merge new references with old data on magnetic tape.

```
ENTER REFERENCE
Anderson GW, et al./J Appl Phys/39(1968)1634/+
AND68A+  Anderson GW, et al.   J Appl Phys        39(1968)1634


ENTER REFERENCE
Anderson GW, et al.   /        J Appl Phys /  39(1968  )1634 /    +
AND68A+  Anderson GW, et al.   J Appl Phys        39(1968)1634


ENTER REFERENCE
ANDERSON GW, ET AL./J APPL PHYS/39[1968]1634/+
AND68A+  ANDERSON GW, ET AL.   J APPL PHYS        39[1968]1634


ENTER REFERENCE
Anderson GW//(68)/+
AND68A+  Anderson GW                              (68)


ENTER REFERENCE
Anderson GW, et al./J Appl Phys/39:3,1634(15Feb1968)/+
AND68A+  Anderson GW, et al.   J Appl Phys        39:3,1634(15Feb1968)


ENTER REFERENCE
Anderson GW, et al./(unknown)/(**)//
AND**A/  Anderson GW, et al.   (unknown)          (**)


ENTER REFERENCE
Anderson GW, Luehrs FU/Journal of Applied Physics/39(1968)1634/Copy
AND68AC  Anderson GW, Luehrs   Journal of Appli  39(1968)1634
```

```
ENTER REFERENCE
Anderson GW, et al./J Appl Phys/39(1968)1634

INSUFFICIENT NUMBER OF FIELDS

INPUT NOT ACCEPTED - REPEAT

ENTER REFERENCE
*Anderson GW, et al./J Appl Phys/39(1968)1634/+

AUTHOR MUST START WITH A LETTER

INPUT NOT ACCEPTED - REPEAT

ENTER REFERENCE
Anderson GW, et al./J Appl Phys/39[3](1968)1634/+

MORE THAN ONE DATE

INPUT NOT ACCEPTED - REPEAT

ENTER REFERENCE
GW Anderson, et al./J Appl Phys/39:3(1968.02.15)1634/+
GW 15A+  GW Anderson, et al.   J Appl Phys        39:3(1968.02.15)1634


ENTER REFERENCE
```

TITLE

Reference Index

UDK 10: List Index

Press to display the index on tape.

If you want a hard copy, enter YES to the prompt.

(This routine does not interfere with the input data.)

TITLE

Reference Index

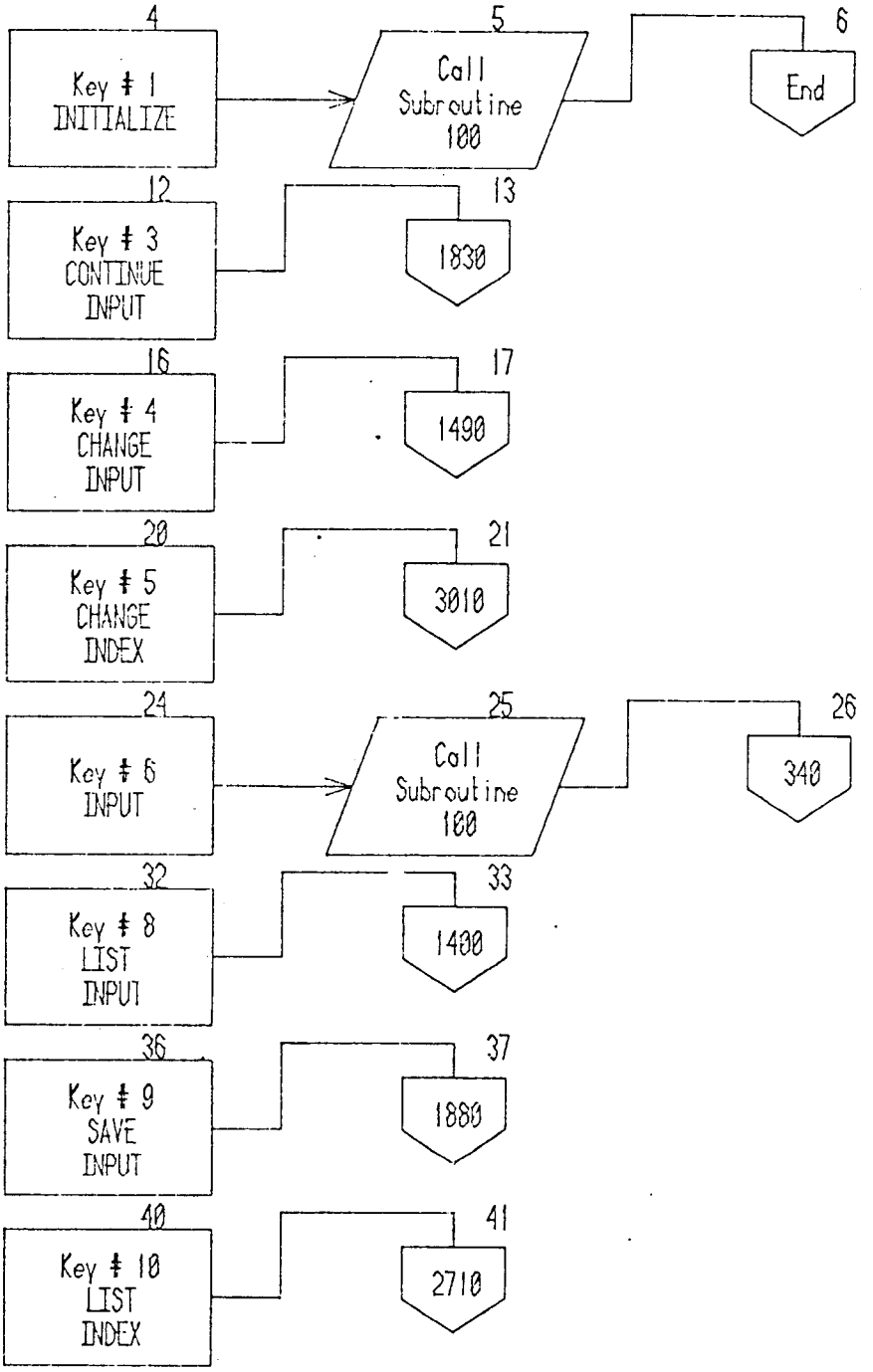
APPENDIX A

FLOW DIAGRAMS

TITLE

Reference Index

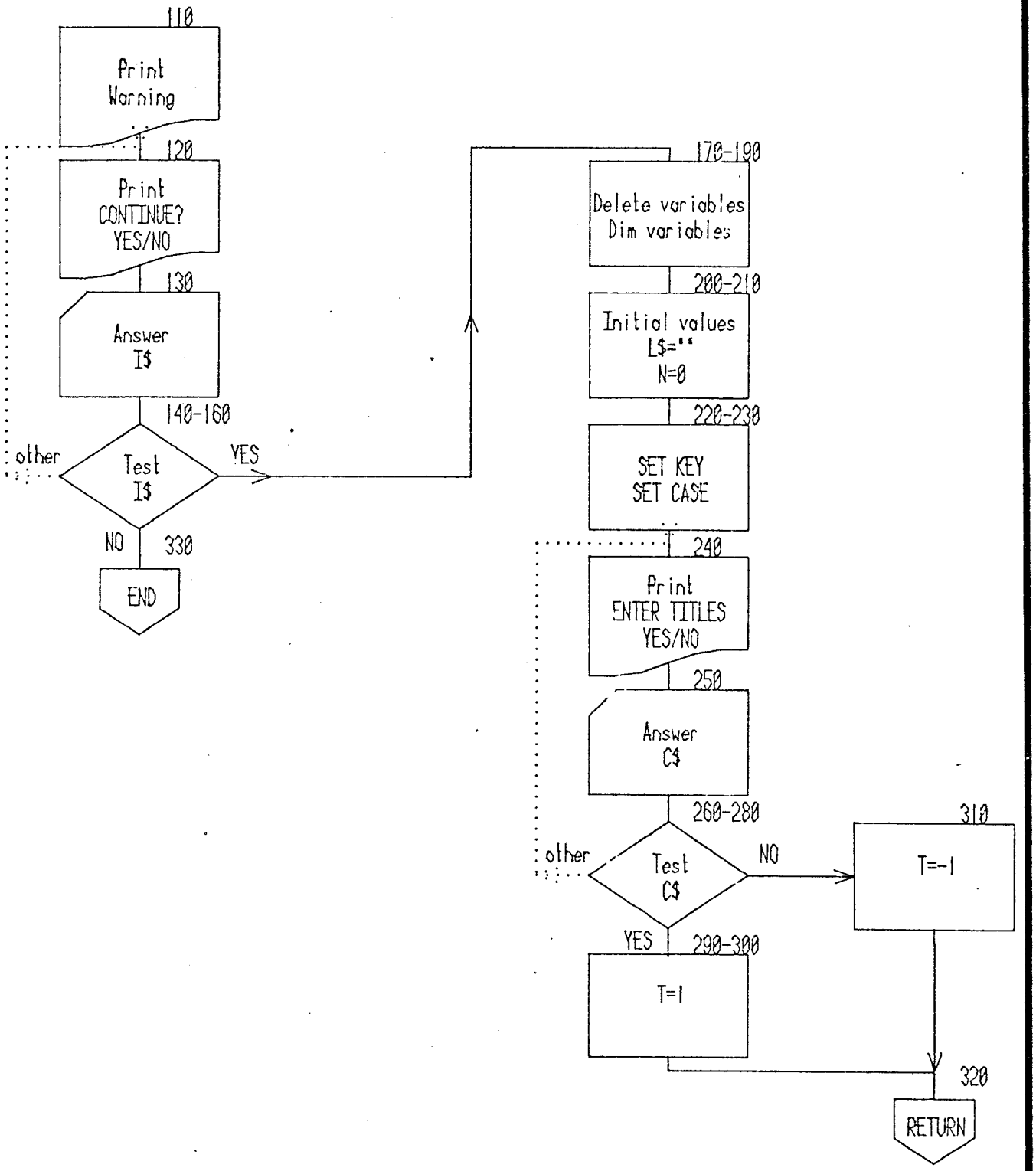
REFERENCE INDEX: USER DEFINABLE KEYS



TITLE

Reference Index

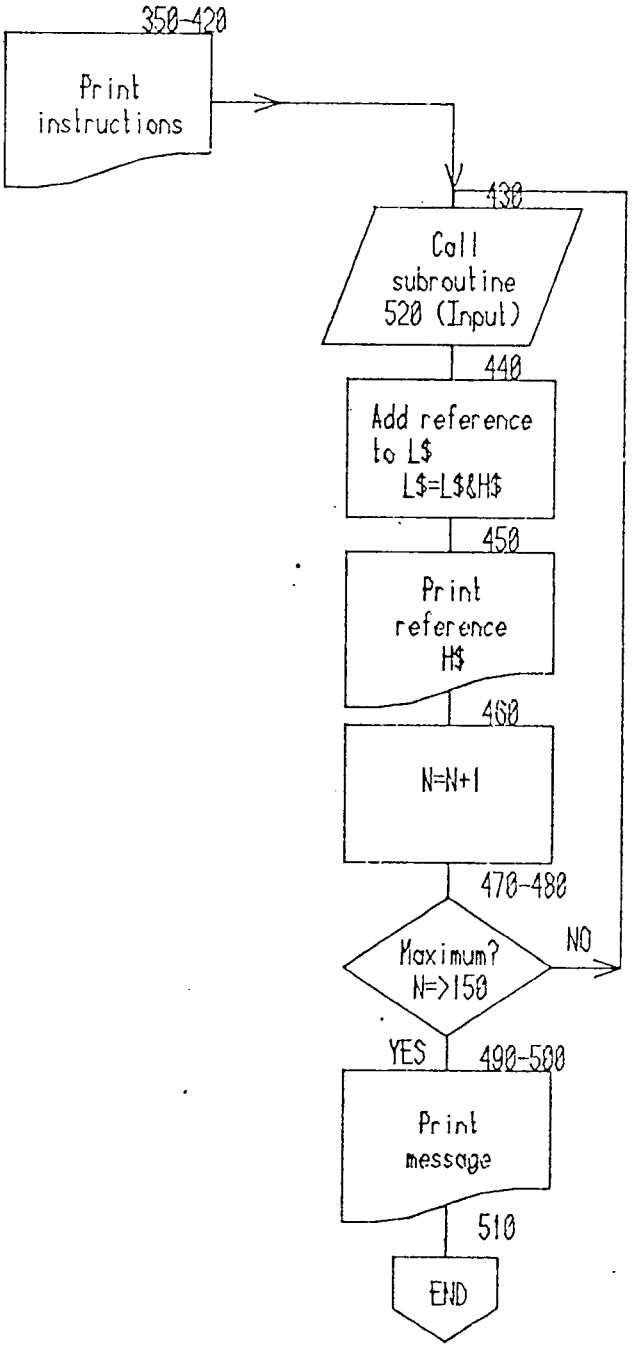
REFERENCE INDEX: INITIALIZE [100-330]



TITLE

Reference Index

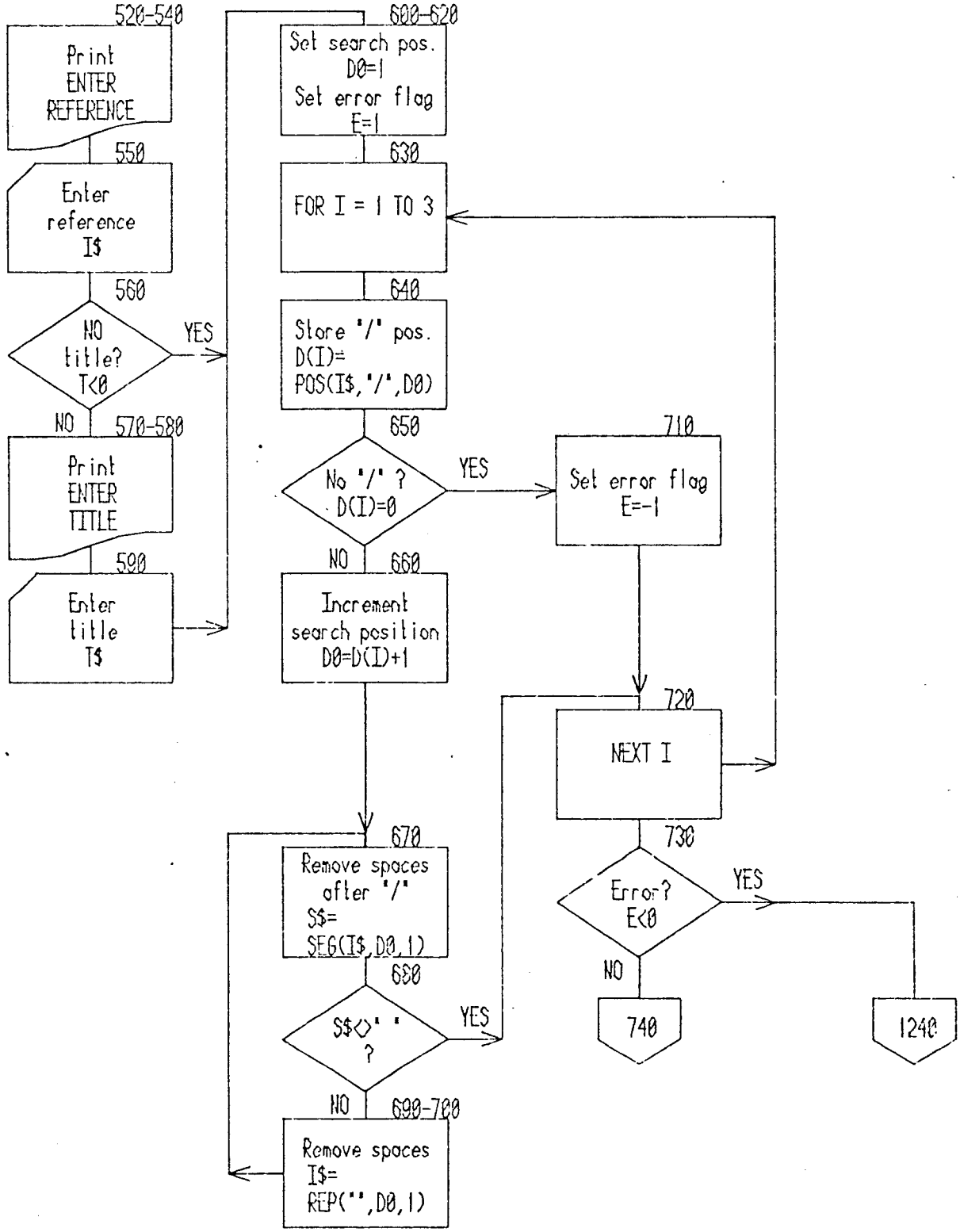
REFERENCE INDEX: INPUT CONTROL [340-510]



TITLE

Reference Index

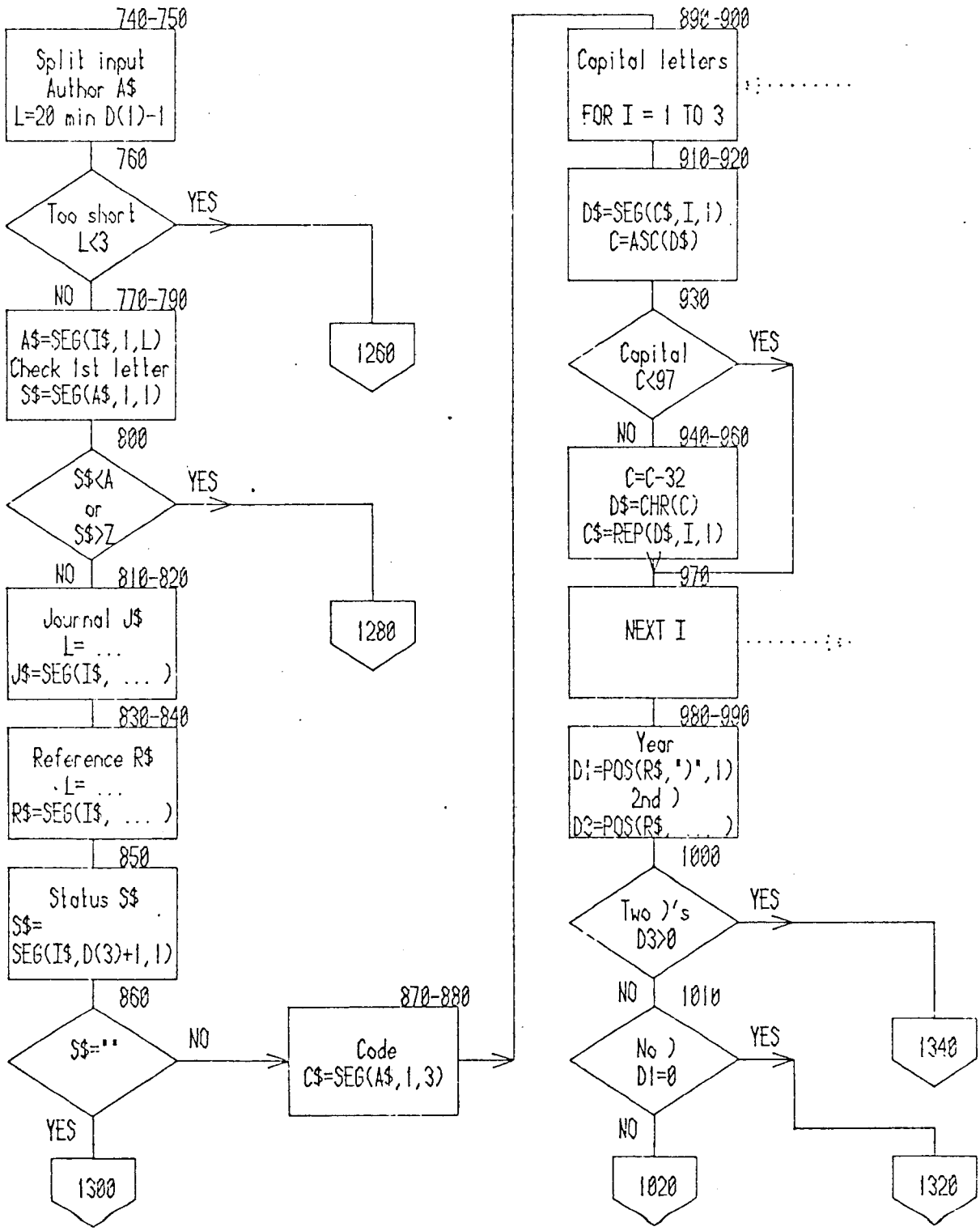
REFERENCE INDEX: INPUT SUBROUTINE (p 1) [520-730]



TITLE

Reference Index

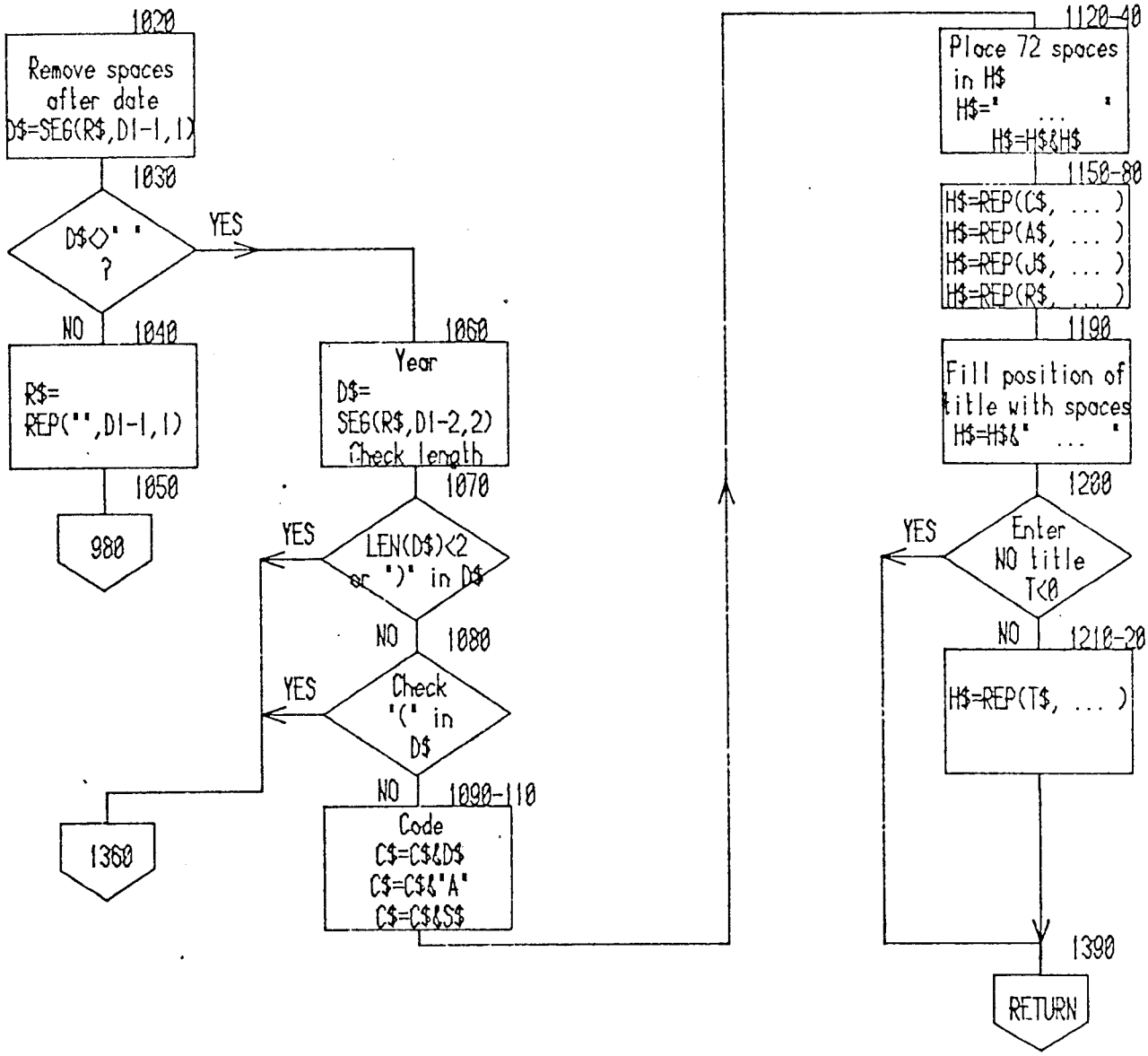
REFERENCE INDEX: INPUT SUBROUTINE (p 2) [740-1010]



TITLE

Reference Index

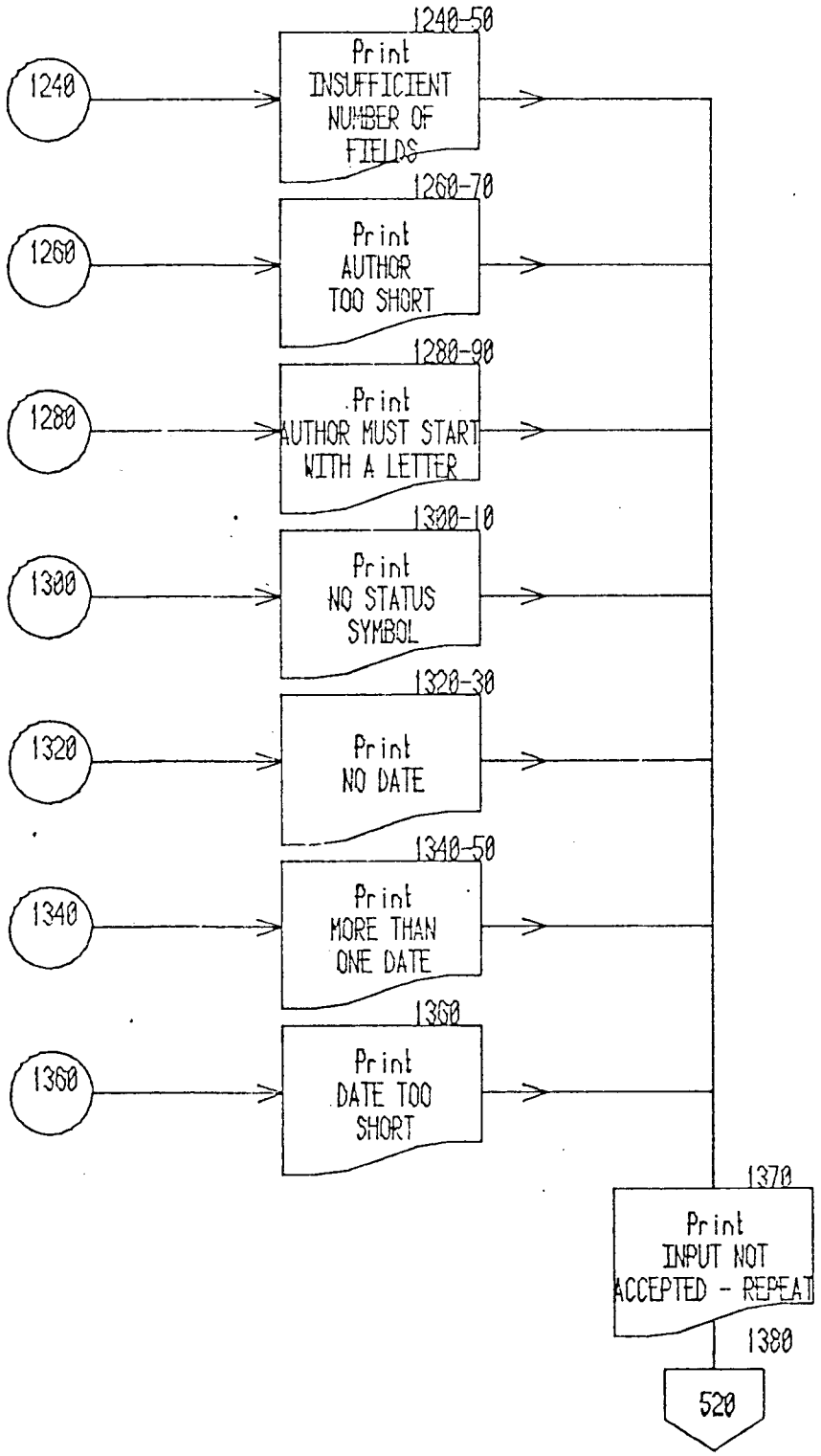
REFERENCE INDEX: INPUT SUBROUTINE (p 3) [1020-220]



TITLE

Reference Index

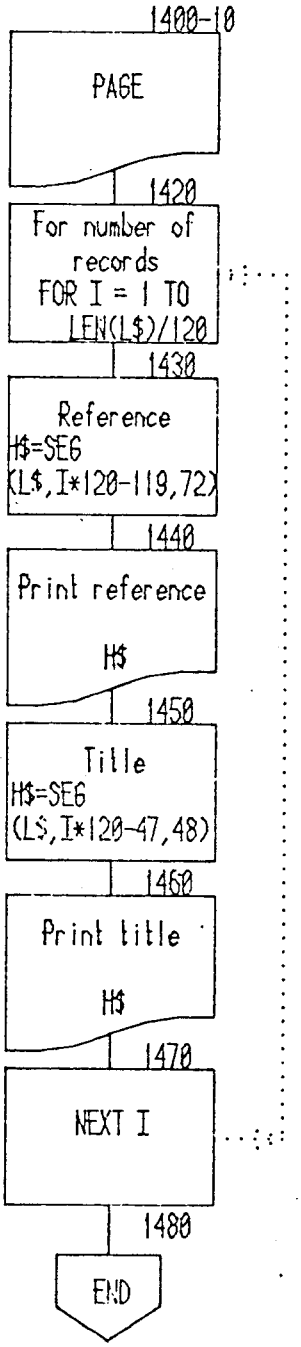
REFERENCE INDEX: INPUT SUBROUTINE (p 4) [1230-380]



TITLE

Reference Index

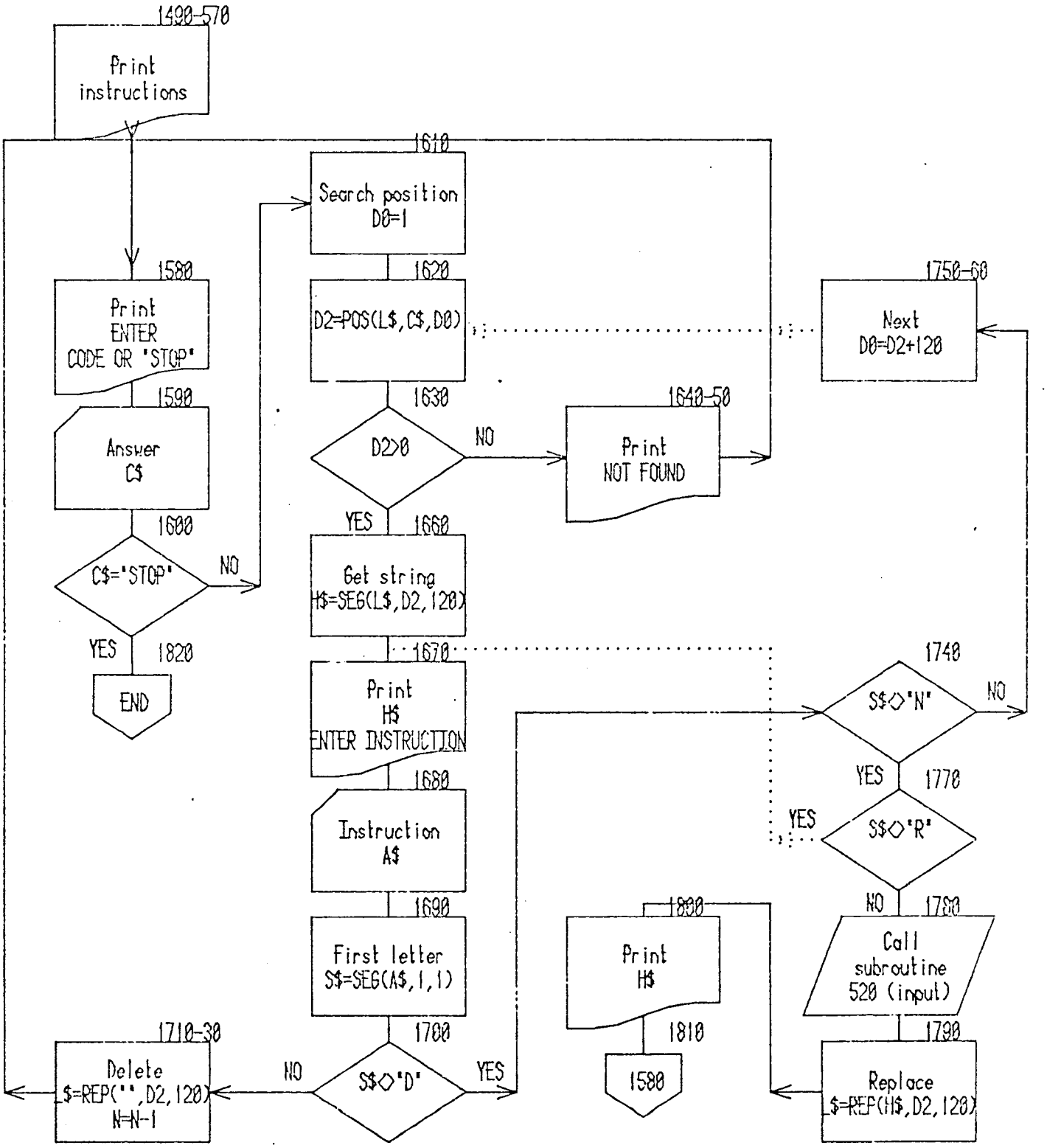
REFERENCE INDEX: LIST INPUT [1400-1480]



TITLE

Reference Index

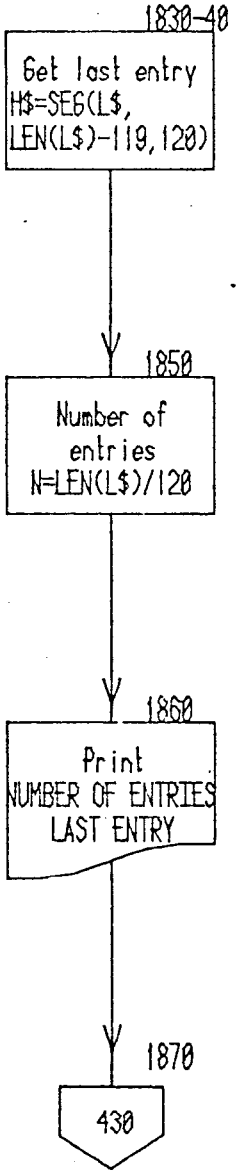
REFERENCE INDEX: CHANGE INPUT [1490-1820]



TITLE

Reference Index

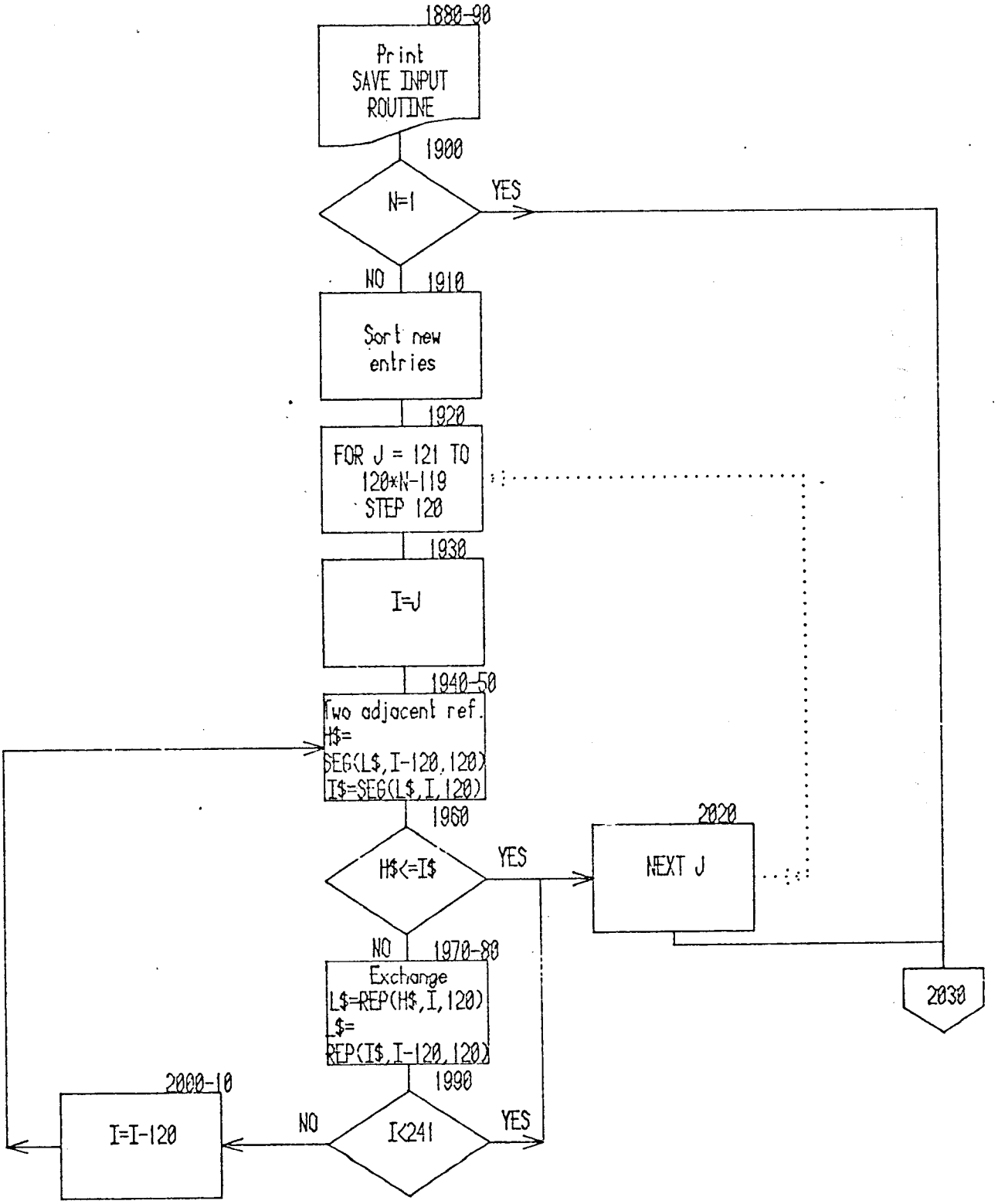
REFERENCE INDEX: CONTINUE INPUT [1830-1870]



TITLE

Reference Index

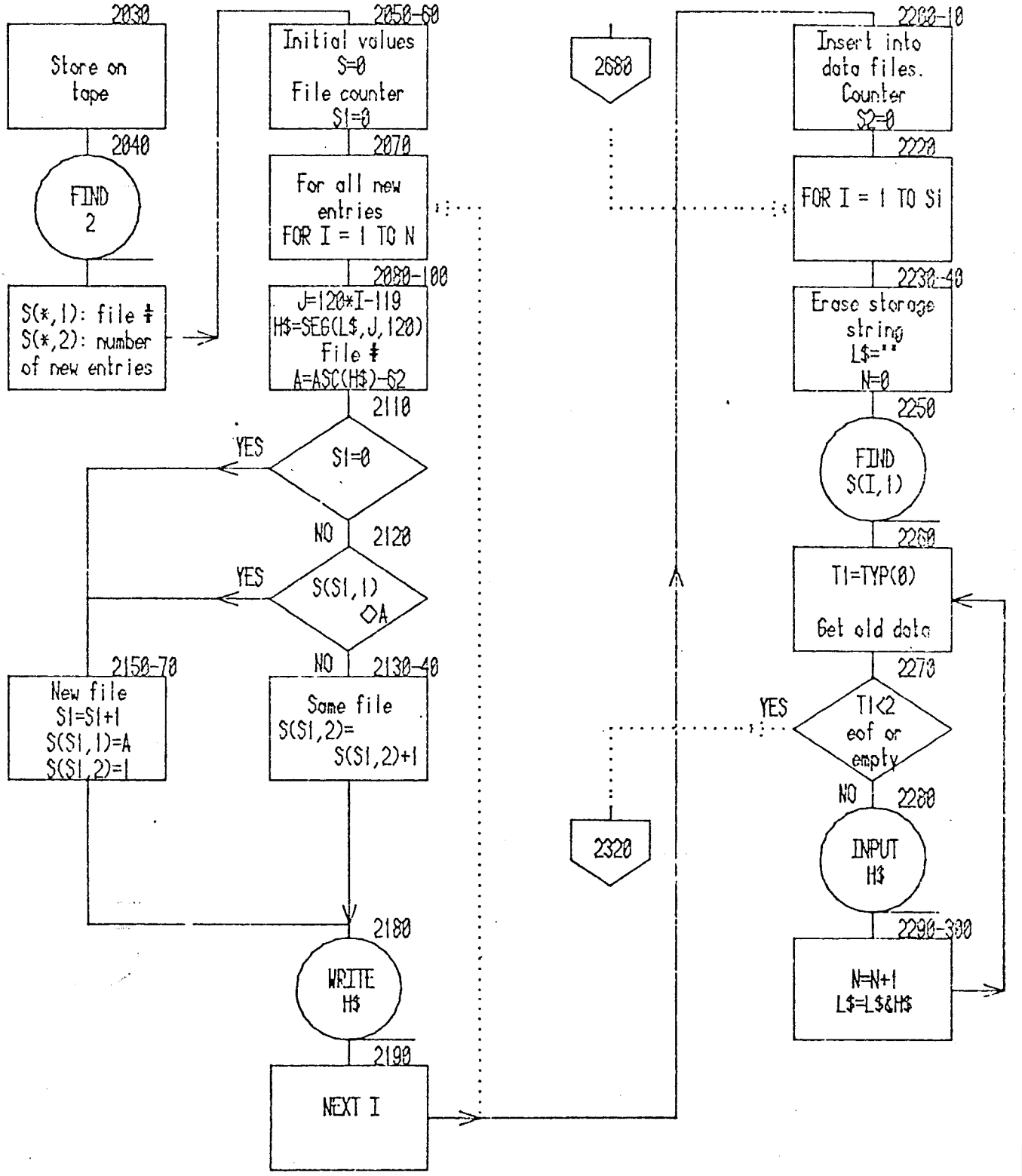
REFERENCE INDEX: SAVE INPUT (p 1) [1880-2020]



TITLE

Reference Index

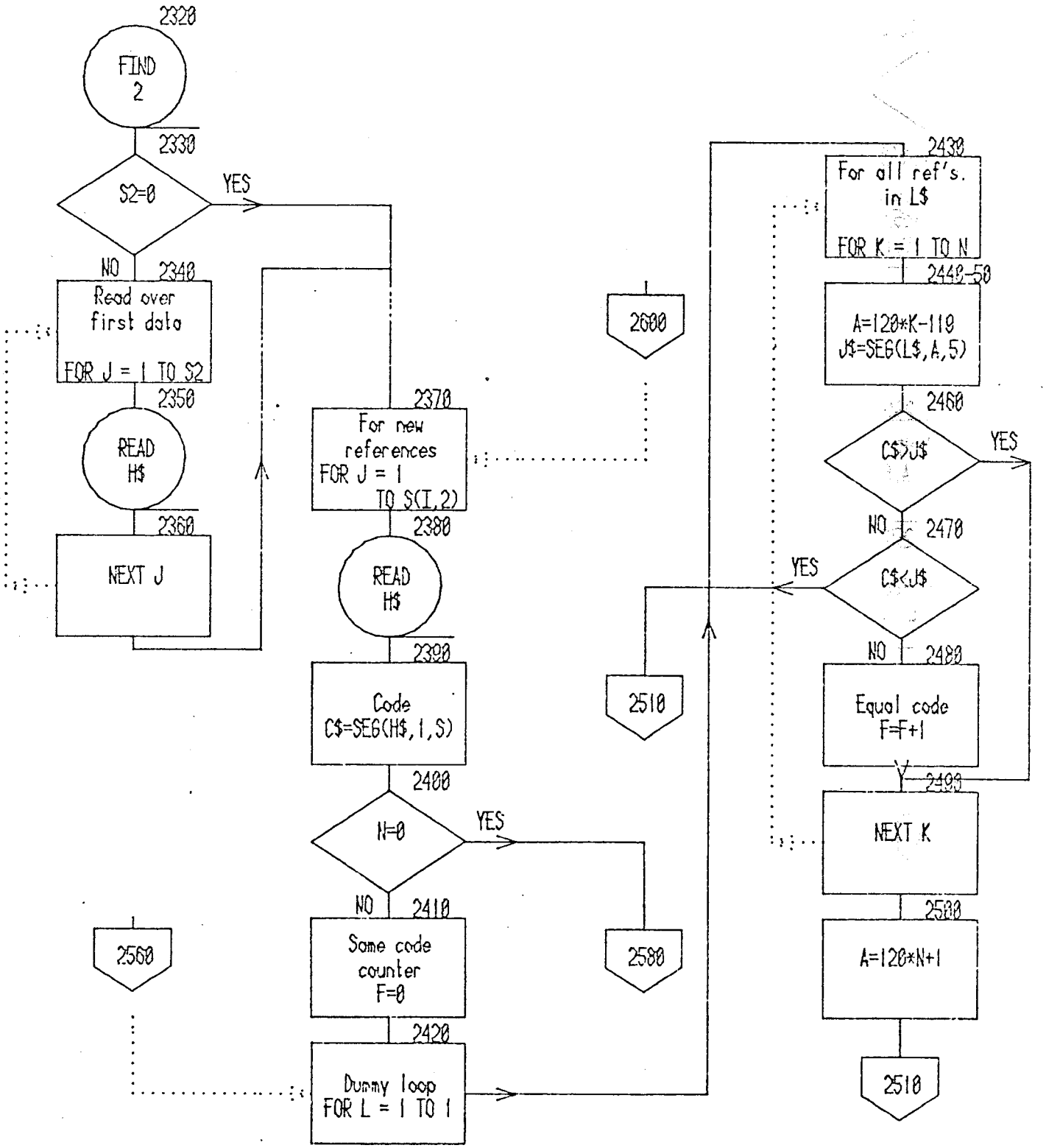
REFERENCE INDEX: SAVE INPUT (p 2) [2030-2310]



TITLE

Reference Index

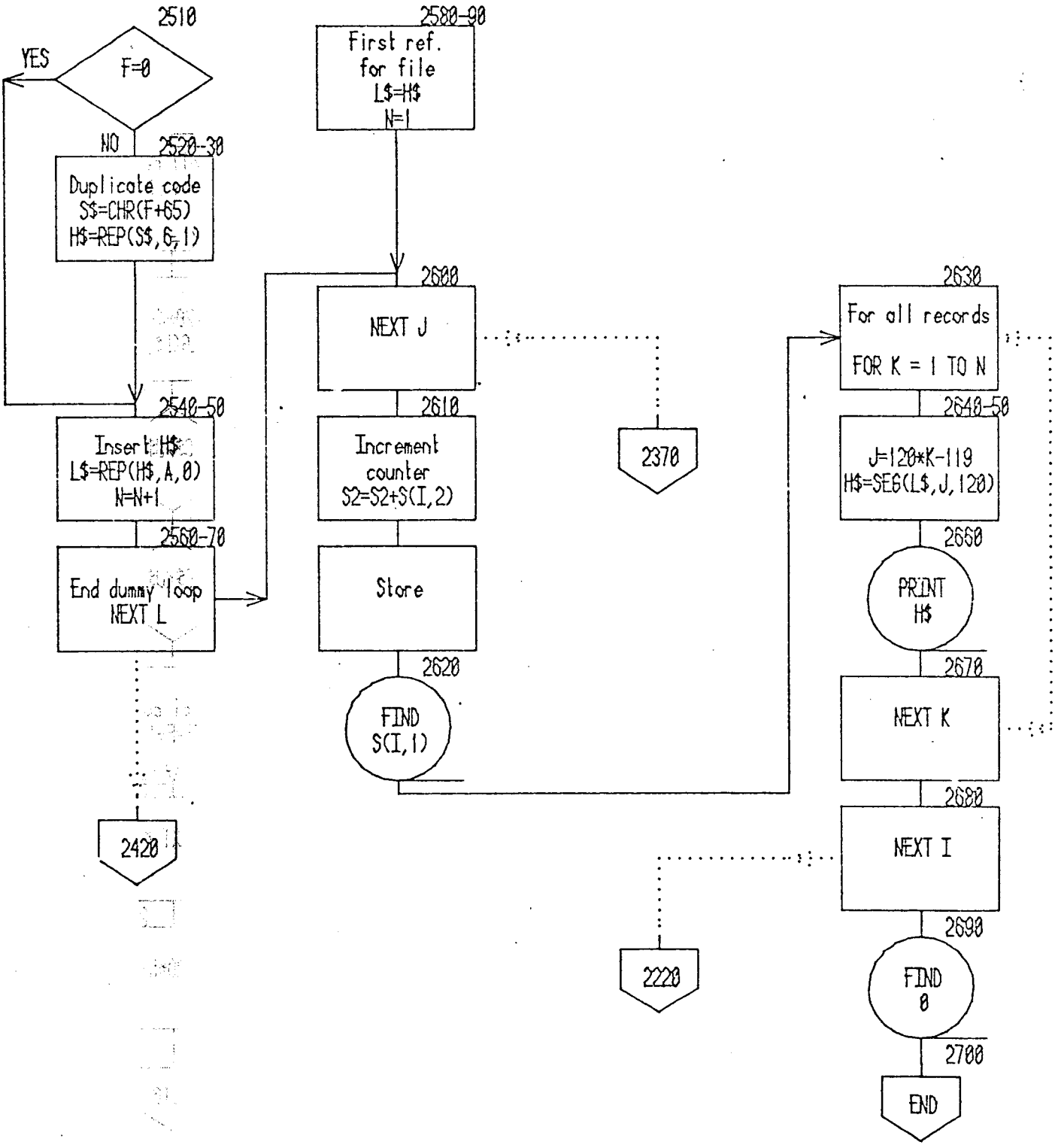
REFERENCE INDEX: SAVE INPUT (p 3) [2320-2500]



TITLE

Reference Index

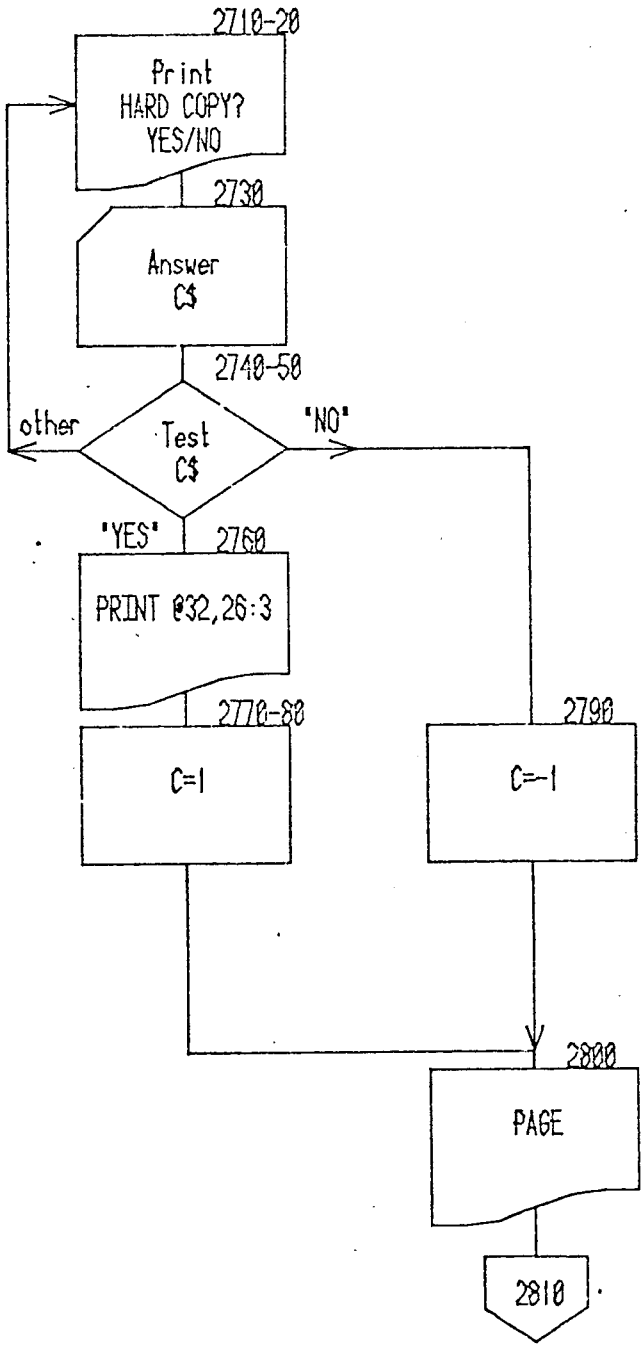
REFERENCE INDEX: SAVE INPUT (p 4) [2510-2700]



TITLE

Reference Index

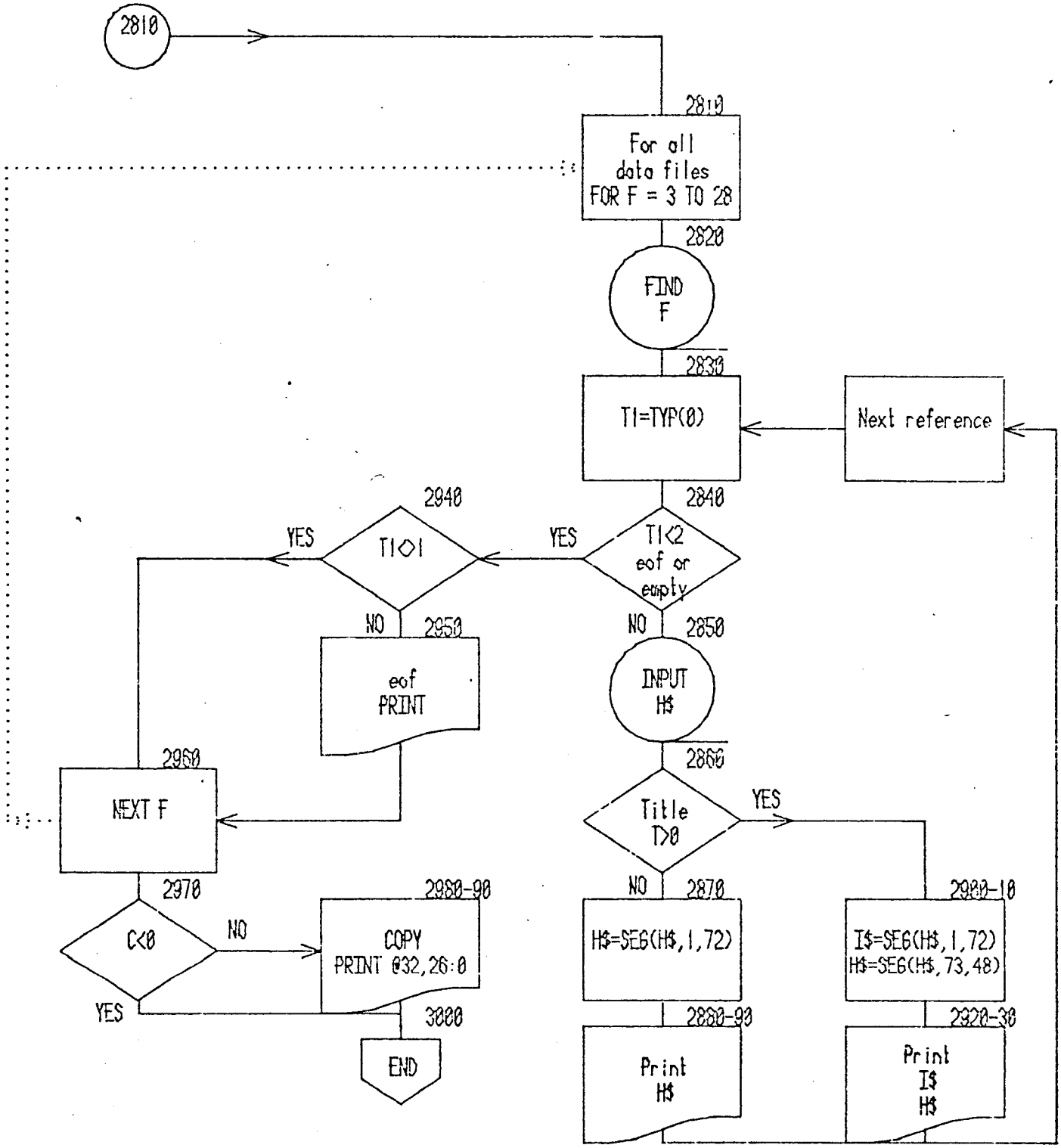
REFERENCE INDEX: LIST INDEX (p 1) [2710-2800]



TITLE

Reference Index

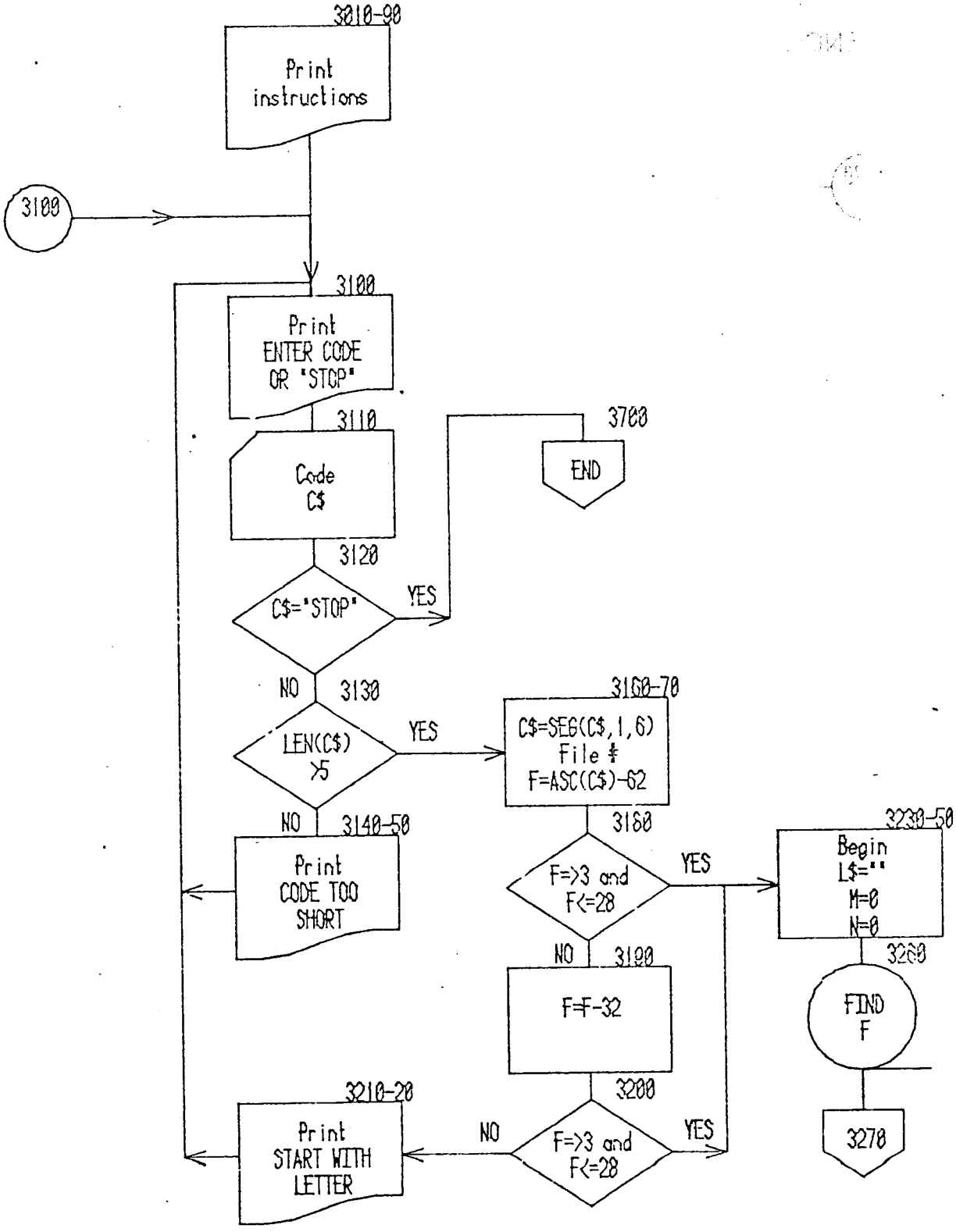
REFERENCE INDEX: LIST INDEX (p 2) [2810-3000]



TITLE

Reference Index

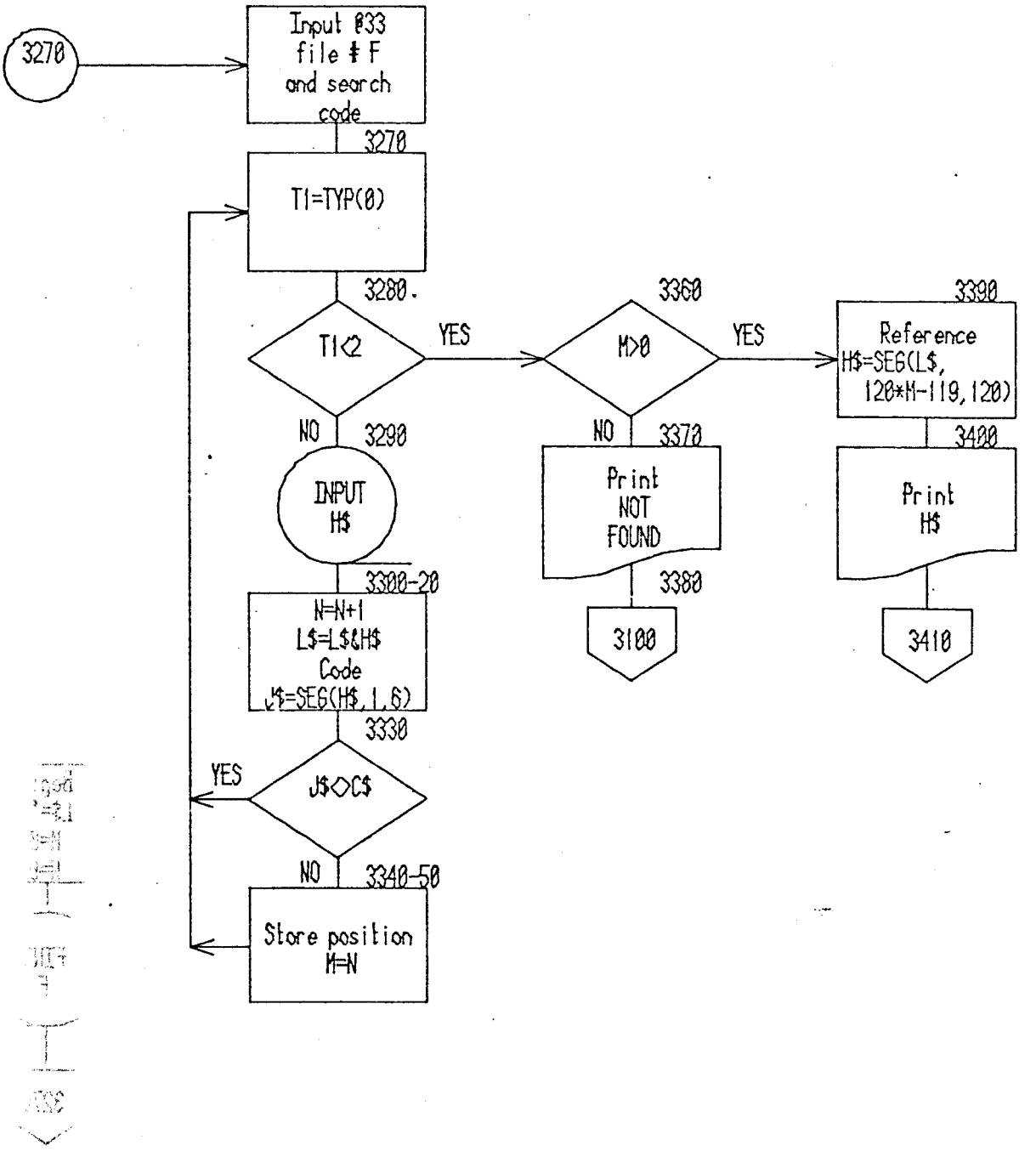
REFERENCE INDEX: CHANGE INDEX (p 1) [3010-3260]



TITLE

Reference Index

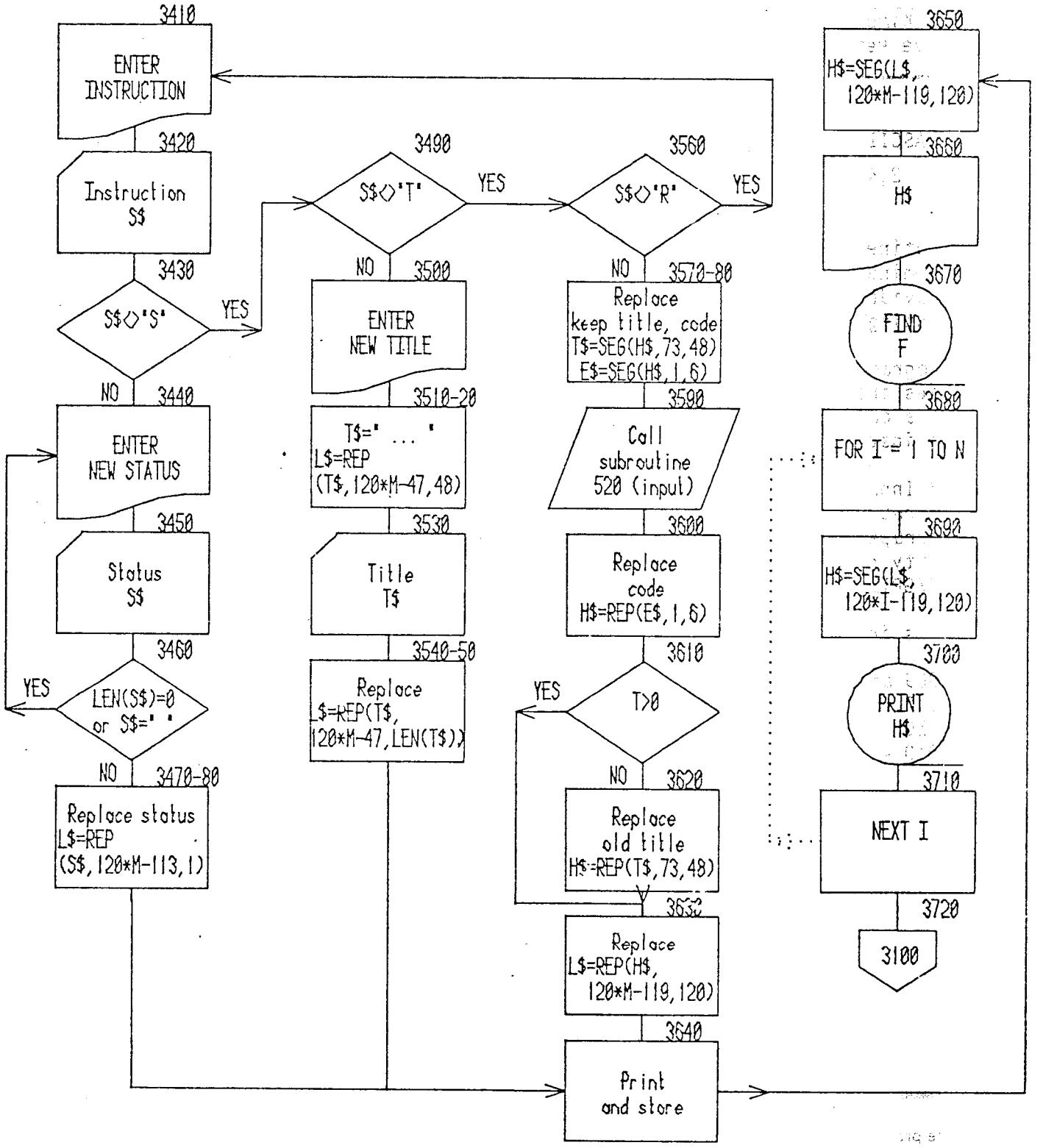
REFERENCE INDEX: CHANGE INDEX (p 2) [3270-3400]



TITLE

Reference Index

REFERENCE INDEX: CHANGE INDEX (p 3) [3410-3720]



Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	
4050/TM5000 GPIB Routines	
ORIGINAL DATE: September, 1981 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: TM5000 Mainframe and Plug-Ins
AUTHOR: Jim Kimball, Barbara Malin and Steve Peterson, Tektronix, Inc.	PERIPHERALS

ABSTRACT

Files: 48 ASCII Programs

Statements: 2,498

These 47 routines, following a directory on file 1, match listings in the GPIB Programming Guide (available separately under part number 070-3985-00). They are provided on this tape to save the user the task of keying in examples from the listings printed in the Guide.

The GPIB Programming Guide aids the users of 4050 Desktop Computers and TEKTRONIX TM5000 series instruments in making the software connection. The Guide introduces the 4050 as a controller for TM5000 instruments, with programming information specific to instrument control. Major topics are:

- GPIB Input/Output
- Interrupt handling
- Interrupt handling statements
- Utility routines
- 4052/GPIB send and receive program

TM5000 series instruments specifically covered in this Guide include:

- DC5009 and DC5010 - Programmable Universal Counter/Timers
- DM5010 - Programmable Digital Multimeter
- FG5010 - Programmable Function Generator
- PS5010 - Programmable Power Supply

TITLE

4050/TM5000 GPIB Routines

PRELIMINARY OPERATING INSTRUCTIONS

Transfer the index and 47 routines comprising this program from the TEKniques Vol. 5 No. 4 program tape to the first 48 files on a tape.

Use one of the tape duplication programs (see page iv of the documentation) to transfer files 46 through 93 from the TEKniques tape to files 1 through 48 on a new tape.

Or, do a TLIST of the TEKniques tape to find the size of each file 46 through 93, MARK a new tape, files 1 through 48 to correspond with these sizes. Then FIND, OLD and SAVE each of the programs.

E.g., After marking a new tape of 48 files,

Insert TEKniques Vol. 5 No. 4 program tape into the 4050

FIND 46

OLD

Remove the TEKniques tape

Insert the new tape

FIND 1

SAVE

Repeat the process for files 47 through 93 on TEKniques tape.

TITLE

4050/TM5000 GPIB Routines

TAPE STRUCTURE

All files are ASCII.

Most are complete programs; some are subroutines.

See the GPIB Programming Guide for listings and instructions to determine whether a given file is a program or subroutine.

DIRECTORY GPIB PROGRAMMING GUIDE EXAMPLES

070-3985-00

TITLE

4050/TM5000 GPIB Routines

DESCRIPTION

File 1: Prints a directory in three columns showing file number, file name (usually the same as a REM statement that is the first line in the listing), and page number of Programming Guide where example is listed.

Files 2-3: Example programs from Section 1.

Files 4-7: Three utilities and a program for 4052 control of a GPIB system from Section 2.

Files 8-18: Program examples for use with the DC5009 and DC5010 Universal Counter/Timers from Section 3.

Files 19-31: Program examples for use with the DM5010 Programmable Digital Multimeter from Section 4.

TITLE

4050/TM5000 GPIB Routines

Files 32-38: Program examples for use with the FG5010 Programmable 20 MHz Function Generator from Section 5.

Files 39-44: Program examples for use with the PS5010 Programmable Power Supply from Section 6.

Files 45-48: Example programs for use with two or more TM5000 instruments from Section 7.

OPERATING INSTRUCTIONS

Refer to the GPIB Programming Guide (070-3985-00) for a discussion of the routines and the listings. Generally, programs in Sections 1, 2 and 7 of the Guide require more than one TM5000 instrument, while Sections 3-6 require only the instrument that is the subject of the section.

Most example programs contain an SRQ handler with only the specified instruments included in the POLL address list.

Other instruments such as a counter on-line while trying to run an example from Section 4 (DM5010) will cause execution to abort if the extra instrument asserts SRQ (as for power-up).

File 1 may be AUTO LOADed to print program directory and page number of Programming Guide where listings appear.

Other files must be accessed:

FIND n } (where n is the file number)
OLD }

RUN or LIST

TITLE

4050/TM5000 GPIB Routines